SYSTEM DESIGN OF A PARALLEL PROCESSING COMPUTER FOR INFORMATION RETRIEVAL

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

[illegible]

BY
T. RADHAKRISHNAN

to the

EE-1971-D-RAD-SYS

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JULY 1971

Th 0 5 4
621.3811
R 118
[illegible]

To

my PARENTS and TEACHERS

CERTIFICATE

THIS is to certify that this work entitled, "SYSTEM DESIGN OF A PARALLEL PROCESSING COMPUTER FOR INFORMATION RETRIEVAL" by T. Radhakrishnan has been carried out under my supervision and has not been submitted elsewhere for a degree.

[illegible]

Dr. V. Rajaraman
Professor of Electrical Engineering
and
Head, Computer Centre
Indian Institute of Technology
Kanpur

Kanpur
July, 1971

ACKNOWLEDGEMENTS

I am deeply indebted to my thesis supervisor, Professor V. Rajaraman for initiating me to the work presented in this report. He has found time and patience for many valuable discussions and has been a constant source of encouragement and inspiration.

I would like to express my thanks to Dr. B. Prasada, Head of the Electrical Engineering Department for his interest in my work and the facilities provided. I am thankful to Professors Harry D. Huskey, H.N. Mahabala, A. Bose and S.K. Basu for the many useful discussions, I had with them and to my friends Dr C.R. Muthukrishnan, Mr M. Ibramsha and Dr. V. Gupta for their help and keen interest in this work. I am grateful to the facilities and the help provided by the Computer Centre staff, in particular to Mr R.N. Basu and Mr S. Kapoor.

My acknowledgements are also due to Mr H.K. Nathani and Mr K.N. Tewari for their excellent typing and to my friends for their help in many unperceivable ways.

- T. Radhakrishnan

Kanpur
July 1971

SYNOPSIS

of the

Ph.D. Dissertation
on

SYSTEM DESIGN OF A PARALLEL PROCESSING COMPUTER FOR INFORMATION RETRIEVAL

by

T. Radhakrishnan

Department of Electrical Engineering
Indian Institute of Technology, Kanpur

July, 1971

This thesis is an investigation of problems in information retrieval and the system design of a computer system well matched for information retrieval.

Performance of retrieval systems, namely, recall and precision can be improved by using multi-attributes for information representation and user feedback about relevant and irrelevant retrieval. An algorithm to select the set of attributes for a data base has been proposed in this thesis. The algorithm is based on the information obtained from statistical co-occurence of attributes in the data base and when applied to a sample data base gives the keywords and citations as the two most important attributes for library information system. The improved performance obtained in feedback retrieval process as compared to non feedback processes has been studied and formalized.

Considering these aspects of retrieval problems a special purpose computer system is proposed. The motivation for this design is three fold:

In information retrieval, matching of documents with the query need not be sequential. This kind of "inherent parallelism" in retrieval problems should be exploited.

There is a hierarchy of operations involved in information retrieval like query analysis, retrieval operation and output presentation, and each of these has its own special requirements. It is desirable to study the feasibility of design of a computer system well matched to each of these levels.

There is an increasing need for computing power at a lower cost due to the rate of information growth and need for improved performance. Also the multi attribute system and feedback retrieval could improve the performance at the cost of computing time.

A computer system with a hierarchy of processors is proposed. This system has a preprocessor operating in a time sharing mode for query analysis and managing the second level processors. The second level of the system has one or more processors which perform the retrieval function and select pointers to the relevant documents. These second level processors have been

economically designed by specializing their function. The output processor in its simplest form can be a soft copy display.

The main difference between this architecture and other array processing machines such as SOLOMON and ILLIAC IV is that inter processor control and communication is only through the central controller. This modularizes the system and endows it with flexibility to meet growing requirements. Performance degradation which might occur due to the lack of direct communication between the parallel processors can be reduced by a suitable file organization technique which minimizes the need for such communication. The proposed architecture is described in PMS and ISP notations which are developed by Bell and Newell to describe computer systems.

A system level simulation of this computer architecture has been carried out with a program written in FORTRAN. System characteristics like arrival rate of customers and computational time required per customer are inputs to this simulator. The number of parallel processors, the number of channels for file access, and the number of output channels are the system parameters. Average waiting time of the customers, response time and utilization of the resources constitute the outputs of this simulator. The results of simulation are used in developing a methodology for the system design. The use of this simulator

general system design of this kind is discussed. The use of an analytical model which supplements this simulation model has been considered.

By making the second level parallel processors highly special purpose a wired in logic for completely hardware oriented search is proposed. A coded representation of the documents and its use in hardware oriented search are discussed. In this method the cost of the parallel processors is no more than the cost of a few gates and registers.

The main conclusion of this thesis is -

It is feasible to design a hierarchical special purpose parallel processing computer system well matched for information retrieval problems. Further it may be made modular to meet the growing needs of an information retrieval environment.

CONTENTS

CHAPTER I

INTRODUCTION

1.1 Need for Information Retrieval

Information retrieval is the selective recall of stored knowledge. The rapid growth of information in recorded form has necessitated the use of automatic computers as a tool for information retrieval. Technical publication is doubling every three years. Abstract journals have been suggested as a means of uncovering the contents of a variety of technical publication and presenting them to the scientific community. Today there are more than ten thousand technical journals and over three hundred abstracting journals. It has been suggested that perhaps the next step should be the creation of an abstract journal that abstracts abstract journals (110)!

It has been estimated that 30% of the efforts spent, in terms of the cost at MIT alone is in redoing things which have been done elsewhere due to the nonavailability of adequate information service centres (63). Besides technical information projects, areas like medicine, government and business have already started using computers to meet the increasing volume of information and retrieval (101).

Over five hundred hospitals were using automatic means in accounting procedures by 1968. Computers are, recently being used to handle medical records of patients. A typical medical information system has more than 1,25,000 examination records and can be accessed on line (101). The MEDLAR system for medical literature analysis and retrieval is a major achievement in this direction (59). It is centered at the National Library of Medicine and planned from the start to be expanded into a decentralized organization. The activities of MEDLARS include indexing, search, maintenance and improvement of the 'Libraries', the thesaurus of Indexing Terms and preparation of a medical glossary.

The government systems for information retrieval include motor vehicle, law enforcement, criminal justice, legislative, and urban planning and control. It has been estimated that motor vehicle applications would typically have over thirty million records. The system will have hundreds of terminals for inquiry by courts, law enforcement and insurance agencies (73). A typical police information network is required to provide response in thirty seconds when inquiry is made to a file of 2,00,000 warrants of arrests (101). In all, there are more than 4,50,000 papers in standard American technical journals and 1,00,000 informal Government reports published every year and this number is steadily increasing (111).

The literature on information systems is almost completely dominated by business applications of large real time information systems. Maintenance and analysis of policy records in insurance area, on-line ticket reservation in air traffic and order-inventory systems are some examples in business application where computers are already in use to handle large volumes of data (101, 68).

The above discussions motivate launching a large-scale program to automatic information retrieval. Bar-Hillel (5) challenges the argument about technical information explosion and questions whether information retrieval is in fact approaching a crisis. From his point of view specialization has been the defensive mechanism that has evolved to combat the growth of technical publications. However, he himself is convinced that the growth of information in the field of law and medicine need special attention. Often the 'information' explosion is stressed more as the major reason for automatic information retrieval, but the need for quick response and man-machine interaction in decision making are not given due consideration (95).

1.2 Information Retrieval System

An information retrieval system has two major sections namely, information storage and information retrieval. The information store is the raw material for retrieval. Information and queries both are normally available in the form of texts. For economical and effective searching, they are represented in an automated system through a set of "attributes" or "keys". Through long experience librarians have found that direct extraction of keys from the original text does not necessarily lead to effective retrieval. (By effective retrieval we mean that almost all relevant documents to a query are retrieved and no irrelevant document is retrieved) Hence it is necessary to modify the attributes of the text to standardize them in some sense (109). When a document is represented by a set of such standardized attributes, it is said to be indexed.

The indexed documents or texts in a conventional library are classified into one or more groups according to a classification rule. The classification rule would normally group all related documents into one group. When a document falls into more than one group, cross references are given to the other relevant groups to which it belongs (89). The main aim of classification is to minimize the effort needed in retrieval. In automatic systems also, the indexed documents are organized into a number of groups and stored in a storage medium, accessible to the machine. This classification and storage in machine accessible form are together known as "file organization". Thus, the raw material for information retrieval process is a set of indexed and organized files.

Then, the information retrieval process can be represented as shown in Figure 1.1. The query records from users for information are first standardized by using the same procedures as in document text standardization. The transformed query is used to "match" with the information items in the store in order to select the set of relevant documents. Often the

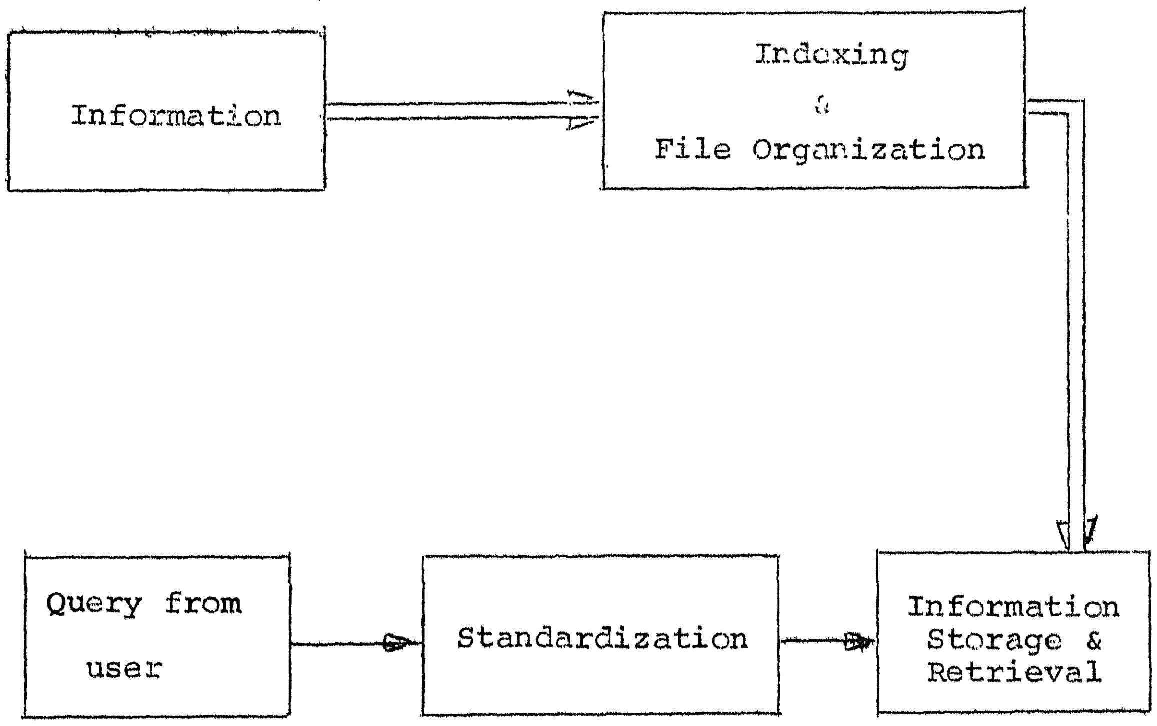


Figure 1.1 Information storage and retrieval scheme

transformed document texts as in the storage file may not be in a readable form for the requester. In such cases the retrieval system selects a set of pointers which refer to their corresponding readable texts, which might be stored in a separate file. Either a soft copy or hard copy of the selected document texts are presented to the user depending on the need and available facility.

In summary, the main components of an information retrieval system are:

(i) an indexed and organized file.

(ii) a retrieval method for matching the transformed query with the documents.

(iii) a query language in which the user states his needs and communicates with the system.

(iv) a set of user terminals with facilities for entering the query and receiving the response.

In a complex information system design these factors cannot be considered independently. For instance, file organization and retrieval method have a number of common factors (109). Similarly the kind of user terminals, user groups and document types in storage are to be considered jointly in developing a query language.

1.3 On line retrieval systems

Batch processing of retrieval requests is simple, but unsatisfactory for a number of reasons (95). Further, the quick response required (in the order of seconds) in military, medical and legal information systems has led to the development of "on-line information systems". The concept of time sharing and interactive processing with third generation computers have made it feasible to design the on-line systems like the one in project MAC (54). SMART system (94) is another example. Licklider (62) in his Libraries of the future envisions a situation in which an user sitting at a desk like inquiry station, connected to an on-line computer system with a large memory store, can call for information, receive it on a cathode ray tube display scope, modify it, and save it for future reference.

The on-line user feedback could be effectively used in improving the performance of retrieval systems. For example, displaying the selected part of thesaurus may be useful in query formulation. Feedback about relevant and irrelevant retrieval would be useful in the retrieval process. The results obtained from various experiments with SMART system have shown that the performance of retrieval systems could be improved over 30% on an average, with user feedback (18).

There are three distinct types of time-sharing installations in practice. The first type (and the one meant more often than not when the term time sharing is used) is the "general purpose, time sharing system" (99). A general purpose time sharing system attempts to provide each user with the full range of capabilities of a general purpose computer. A second type of system sometimes goes by the name "Dedicated system".

Here the user is constrained to use but a single language (29). The third class of system is the one in which the language of the user is further restricted and specialized to deal with a restricted set of problems. The SAGE air defence system and the SABRE air line reservation system are some examples of this kind (80).

The need for sharing a common data base like technical literature, medical data, legal information or business data among the wider group is constantly increasing. Nationwide information networks and international information transfer have been projected (79). This thesis is the study of a special purpose computer system for information retrieval which will be useful in the various application areas mentioned above.

1.4 Motivation for a Special Purpose Computer System

The early development of computers and the resulting applications have been in the numerical computation of functions. Developments in procedure oriented languages (46) and operating systems have enhanced the number of users of this facility. Hardware developments in areas like semiconductor technology for logic and core technology for memories have made the design of large computer systems feasible for this purpose. Hardware developments have contributed not only to the improvement in reliability of operation of such a large system, but also towards reducing their cost to acceptable limits (104). As the number of computer installations increased, they began to be used for solving problems in many areas other than numerical computation. Applications of computers in non-numeric problems (which basically do not involve numeric calculations) like picture processing, information retrieval and pattern recognition are some examples (36). As a result special program packages and problem oriented languages have been developed for this purpose (97). All these attempts have been in developing software means to use the machine, which is designed for numeric computation in solving non-numeric problems. Attempts in designing a computer system for non-numeric applications which would be structurally different from a numeric machine were rare. The main reasons for this could be, the imbalance between the cost of such a system and its demand which existed in the last decade. Further, the cost of software was a small fraction of the hardware cost of computer systems.

The advent of integrated circuits and their application in computer industry have changed the imbalance between hardware and software costs (35). The hardware cost is showing a decreasing trend whereas the cost of software is constant or even increasing. The hardware cost of C.P.U. has been estimated to be three percent of the total cost of recent systems (4). Semiconductor memories costing 1.5 cents per bit have been projected with developments in LSI (98). As a result of these developments, computer systems with more than one processor and memory (unlike the conventional Von Neumann's uniprocessor computers) have been conceived.

A parallel processing system with as much as 256 processors, each with 4K memory has been designed at the University of Illinois for solving problems in partial differential equations and weather forecasting (7).

Having been motivated by the feasibility of such special purpose system design, this thesis considers a computer system well matched for information retrieval applications. This study is further supported by the increasing applications of computers for retrieval of information not only in library information systems but also in areas like, military information system, medical, Government, business and legal information systems.

The special purpose computer organization discussed in this thesis considers the following special nature of information retrieval problems. The problem of information retrieval has an "hierarchy of functions" to be performed in a sequence, namely, query pre-processing, selection of relevant documents to the query and display of the selected documents to the requesters (See Section 2.4). The architecture considered in this thesis has three levels of processors well matched at each level for the function being performed. Secondly, information retrieval problems have "inherent parallelism" (see Section 2.4) which can be exploited by providing more than one processor to operate in parallel. Each of these parallel processors will be restricted in their operations to handle particular data types. This would reduce the complexity and the resulting cost of such processors.

The proposed architecture with a set of processors operating in parallel has been simulated on IBM 7044. The use of this simulator in designing a computer system to meet the given requirements of an information system like the arrival rate of customers and desired response time from the system, is presented. This simulation shows that by increasing the number of specialized parallel processors it is possible to get a response time of the order of a few minutes as required in on-line information systems (73). Such computer systems will be useful in feedback information retrieval systems where the on-line interaction with the requester is used in the retrieval of desired information.

By restricting to a particular indexing and coding scheme, it has been shown in this thesis, that the cost of a typical parallel processor of the kind mentioned earlier is essentially governed by the cost of a few gates and registers.

As the retrieval operation in this architecture is done by a separate set of processors operating in parallel, the system is modular. When the work load increases (in terms of number of matching required) it is possible to maintain the same grade of service (in terms of response time) by suitably increasing the number of parallel operating elements in the system. This increase in work load may be an outcome of increasing volume of information to be handled or the use of special techniques like multiple attributes for information representation in order to improve the performance of the retrieval system.

1.5 Outline of the Thesis

In Chapter II, a brief review of the literature is presented. As this thesis is a discussion of a special purpose computer system for information retrieval, the review is divided into two sections. The literature on information retrieval system is vast. A brief discussion of the various problems in information retrieval and the results in each area have been presented. The second section is an over view of the parallel processing systems proposed in the literature for various applications. A proposal for a special purpose computer system for information retrieval with due considerations to the parallelism in the problem and hierarchy of operations involved is given.

Chapter III is a summary of some results obtained in multi-attributes in retrieval process and user feedback as means of improving the performance. The contents of this chapter when considered with the growing volume of recorded information have formed the motivation for the special purpose computer system considered in this thesis.

The details of the proposed computer architecture are presented in Chapter IV. This architecture has a hierarchy of three processors in the second level of which there can be more than one processor operating in parallel. The notations developed by Bell and N$\overset{e}{ø}$well (12) in describing a computer system have been used to describe the proposed architecture. The advantages of this system and a typical instruction set for the second level processor are given. The need for considering the first and third level processors as special purpose time sharing systems in reducing the cost of the total system has been discussed.

Chapter V presents the details of a system level simulation of the proposed architecture. This simulator has been written in FORTRAN IV. The common problems involved in the simulation of a multiprocessor system and concurrent events are discussed. A brief description of theSIMULATORand a methodology in using the simulator for system design are considered. A number of possible modifications for the simulator to meet the special conditions are indicated. A discussion about formulating the system design problem as a mathematical programming problem and its shortcomings are pointed out.

In Chapter VI, the SUPERIMPOSED coding scheme developed by Huskey and Files (32) is used. With this coding scheme and a parallel read disk, it has been shown that the second level processors in the above architecture are much simpler. In fact, the cost of a typical second level processor, in that case is the cost of a few gates and registers. The price, for this reduced cost is in terms of the flexibility and performance of the retrieval system.

CHAPTER II

REVIEW OF LITERATURE AND STATEMENT OF THE PROBLEM

2.1 Introduction

The term "information retrieval" was first coined by Calvin N. Mooers in the year 1960 at a Computer Conference (74). Since then there are a number of activities and the term itself has assumed different connotations. The main reason for these varied connotations is the difficulty in defining the term "information" in a general context. Attempts have been made to use Shannon's (103) definition of information with little success. The semantic value and the form of information are also involved here, rather than the mere uncertainty which determined the information content in Shannon's theory. As a result, a number of interpretations to information retrieval exist in the literature and the most commonly used ones are given below.

An information retrieval system which retrieves complete texts of documents or document surrogates like abstracts are commonly known as "document retrieval system" (58). A system that provides only citations of relevant documents is "reference retrieval system". A system retrieving words or numbers as response to a query is called "data retrieval system". For example a manufacturing company may have a data retrieval system to display on request its sales records, inventory items and personnel files etc. Such data retrieval systems combined with other management functions have come to be known as "management information system" (1). A fourth type called "fact retrieval systems" or "question answering systems" deal with answers to specific questions (110). For example, "what is the area of the triangle whose sides are x,y,z units long?" is answered by such systems. But their area of discourse or subject coverage is very much limited as compared to document or reference retrieval. For instance, a typical question answering system covers topics on high school algebra (16) whereas a keyword system could cover wider areas like computer sciences or electrical engineering and so on. However, document retrieval and reference retrieval are the two common usage of the term "information retrieval".

In all these information systems, we have the system containing the organized information on the one side and the users with queries on the other side. Thus it is possible to picture the activities of information systems in an unified way, as shown in Figure 2.1.

The following are the main aspects in the study of an automatic information retrieval system:

(i) Automatic indexing and abstracting
(ii) Automatic classification
(iii) Semantic problems
(iv) Retrieval methods and file organization
(v) Dissemination of information
(vi) Evaluation and feedback
(vii) Retrieval models, and
(viii) Information networks.

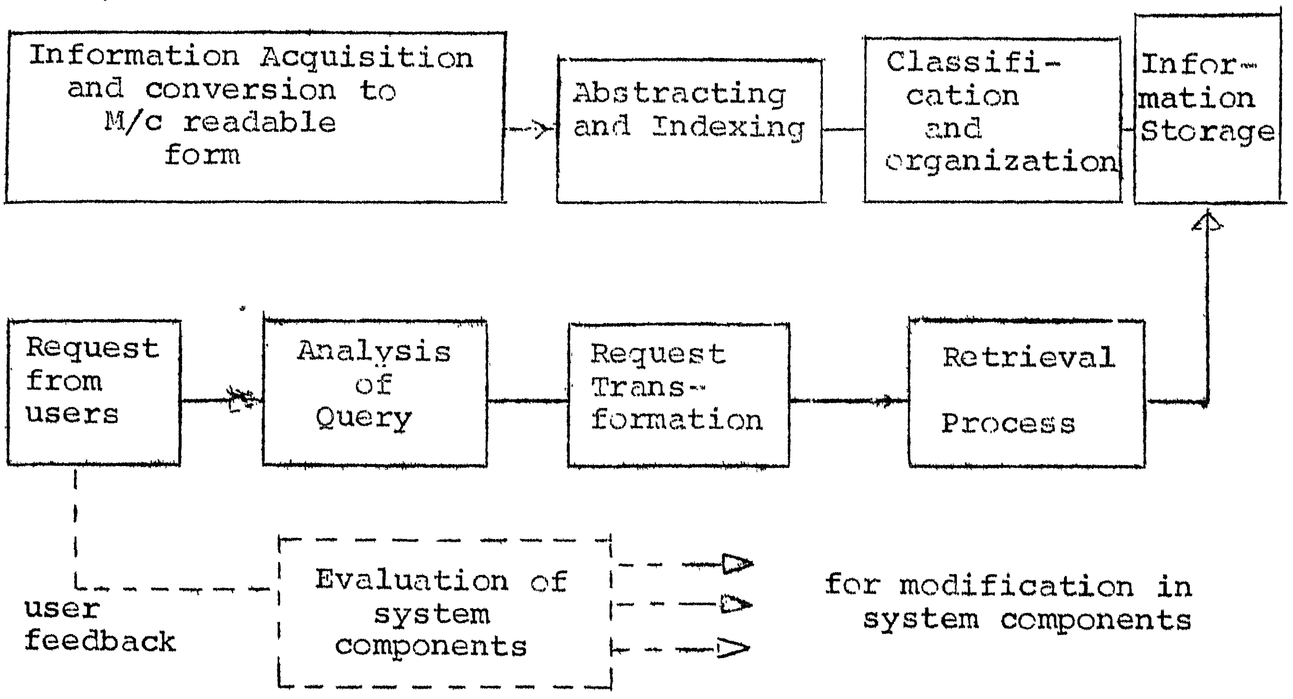

Information Acquisition and conversion to M/c readable form
Abstracting and Indexing
Classification and organization
Information Storage
Request from users
Analysis of Query
Request Transformation
Retrieval Process
user feedback
Evaluation of system components
for modification in system components

Figure 2.1: Information Storage and Retrieval cycle.

2.2 Information Retrieval - A Brief Review

The state of art in the above mentioned sub-areas of an information retrieval system are briefly discussed below:

Automatic Abstracting and Indexing

Indexing is the process of selecting the attributes of the information item. The performance of a retrieval system depends on indexing and should cater to the needs of the majority of the users.

There are many kinds of indexing (9) in practice. In coordinate indexing, each item of information is represented by a set of keywords. The selection of key words is normally controlled by an "authority list" of keywords. In a dynamic system this list needs constant modification. In this system, only keywords are represented but not the relation between them. Often keyword systems with tree and graph structures are used to account for the relation between keywords (96).

The citation index (37) using the bibliographic coupling between documents is another tool for information retrieval. The bibliographic coupling used by Kessler (53) is based on the number of common citations and the number of co-citations. Maron and Kuhns (66) have defined yet another indexing called "probabilistic indexing". In this, each index term is associated with a relevance number in the interval (0,1), which indicates the probability of that index term representing the document.

The problem of abstracting is one level higher in the hierarchy and representative sentences are used to describe the document content, rather than the keywords. Automatic abstracting and indexing methods (30) use the relative frequency of keywords. Automatic abstracting, using the introduction and conclusion paragraphs have been found comparable with manual abstracting (52). Index terms given by the authors in the publications like IEEE and ACM will be a good starting point for automatic indexing, though it is not complete by itself. An automatic scheme for indexing and abstracting is more consistent when compared to the manual methods and avoids undue delay.

Automatic Classification

There are two general types of classifications, namely, enumerative classification and derived classification (58). The enumerative classification has fixed number of categories and the documents are assigned to one or more of these categories. It can be a non faceted classification like Dewey Decimal or faceted classification like Ranganathan's colon classification (90). In faceted classification, the depth of classification can be to any extent by repeated application of the classification rules and is more suited for automatic means.

In derived classification, the classification groups are derived automatically from the data base (67). Statistical relation between the documents in the data base are used for this purpose (17, 3, 38).

Syntax and Semantics Problem

The two requests regarding "transport from Russia to Cuba" and "transport from Cuba to Russia" have the same set of keywords which causes unwanted retrieval when only one of them is required. The relation between keywords in the form of directed graphs was first used by the SYNTOL group (27) to avoid such problems. This was later modified and called criterion phrase by Salton (96) in his SMART system.

By semantic problem, we mean the semantic equivalence of descriptors. Construction of the dictionary called thesaurus, giving the semantically equivalent descriptors for a given descriptor is a challenging problem. User interaction in determining the validity of this equivalence is proposed as a part of an interactive retrieval system (18). Another problem which occurs due to non-standard word ending, however, is not considered to be a semantic problem. For example, computer, computers and computors all will have to be treated as equivalent. The common word stem obtained after deleting the non standard endings like "ed", "ly" is normally used for this purpose.

Retrieval Process and File Organization

A document is retrieved as a response to a query when the relatedness between them exceeds a certain threshold. There are a number of relatedness measures used for this purpose (9,50). The retrieval process and file organization are closely related. When all the documents of the file (or sub files) are to be scanned, a sequential file would be appropriate. A co-ordinate indexing with keyword matching on the other hand would need a semirandom access file like magnetic disk. The sequential, random access and combined file organization have their own advantages and disadvantages (22). The choice of file organization is governed by:

(i) cost of storage and storage efficiency,
(ii) cost of updating and its frequency,
(iii) the retrieval process; and
(iv) available storage device types -- like disk, tape, etc.

Another kind of information organization and retrieval stems from the associative or content addressed memories (86). Processors based on such memories are referred to as "associative processors" (60). As the cost of associative memory is high, these processors have them in a limited size and the file is processed in chunks. Each chunk can be handled by a content addressed memory of appropriate size.

Information Dissemination

The user terminal which provides the communication between the user and the system is important in an interactive retrieval system. In fact, retrieval systems (105) and the question answering systems (91) have such terminals as an integral part of the system. CRT display for soft copy output with a keyboard for input is commonly used.

Unlike the service on demand system, H.P. Luhn of IBM has suggested a "current awareness system" (64). This is also known as selective dissemination of information (SDI).

In SDI service, the interested users are permanently in the system, represented by the "user profile". As new documents are added to the system, the interested users are periodically notified. An interesting feature of this system is the user feedback. The performance of the retrieval system can be evaluated and the user profiles modified to suit the needs of the users. This is possible because the user profile is kept permanently in the system.

Performance Evaluation

The performance of an information system can be considered from the point of view of users or system designers. The two performance measures suggested by Cleverdon (24) are recall and precision and continue to be the commonly used measures. They are defined below:

$$\text{Recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents in the system}}$$

$$\text{Precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

From the above definitions it is obvious that the recall would need the knowledge of the total number of relevant documents and precision would need user judgement. Further they depend on the exhaustivity or specificity of indexing. Thus a set of queries, their recall and precision with different indexing schemes would be a measure of the indexing system itself (24).

Various experiments have conclusively shown that there exists an inverse relationship between recall and precision for a fixed indexing scheme. Lancaster has verified the same with the data base obtained from MEDLARS (59) and classifies the users into groups who need high recall and acceptably low precision and vice versa.

Attempts have been made to use Shannon's formula for information transmitted over a noisy channel for the evaluation of information retrieval process (70).

Interaction between users and the system at the following levels have been suggested as means of improving the performance:

(i) In search formulation and instructing the user in the use of query language.

(ii) In determining the synonyms of the keywords used.

(iii) In examining the partial search result and controlling the search process.

(iv) In controlling the precision-recall trade off by making specific or generic queries.

However, the performance of the retrieval system from the management point of view is not given much consideration except by Lancaster and Mueller (76).

Retrieval models

Becker and Hayes (9) have studied the problems of information retrieval using Boolean algebra and linear algebra. The quantitative studies by Morse (75) of the library system uses the notions of probability theory, queuing theory and Markov process to model book ordering, circulation and other activities of a library. Assuming the documents as a collection of points in an n-dimensional metric space and using barycentric coordinates, Hayes (42) is able to describe the information retrieval process. Salton has used a set theoretical model for information retrieval and studied the inverse relation between recall and precision (96).

The mathematical models developed in the literature are mostly intended to describe existing information retrieval systems. Results obtained from them do not seem to have any substantial impact on the systems being designed.

Information Networks

The inter library loan system is one step towards information networks in manual systems. The project INTREX at MIT (79) and th article by Kemney (51) discuss nationwide and international information networks. The computer networks (10) project undertaken at CMU as the part of ARPA project will have much relevance in this context.

In summary, the trend in recent experiments and literature show that keyword matching and directed graphs are popular tools for retrieval. User interaction and feedback in retrieval are being considered with the recent time sharing systems. The evaluation of retrieval systems still uses recall and precision measures and the theoretical models are not adequate enough to influence the practical or experimental system designs and need deeper study.

2.3 Parallel Processing Computer Systems - A review

Computer systems with special architecture for applications in various fields have been considered in the literature. Computer systems have been developed for airtraffic control (45), weather forecasting (102), solving partial differential equations (14), picture processing (15) and space missions (57). In this section a brief discussion about the parallelism in data processing and a review of computers designed to exploit such paralleli are presented.

Originally the term parallelism was used for parallel transmission or processing of several bits of a word. Recently the ter is used to include any situation where two or more functions are performed at the same time (92). There are two kinds of parallelisms defined by Lehman (61): (i) macro parallelism and (ii) micro parallelism. When several units of a multi-processing system are utilized to process, in parallel, independent sections of a job, we exploit the macro parallelism. On the other hand when relative independence of individual machine instructions are exploited as in the case of lock ahead machines we call it micro parallelism. A parallel processing may be considered to be one in which macro parallelism is used. By convention the parallel execution of I/O is not termed as parallel processing.

Flynn (34) distinguishes between the following two kinds of macro parallelism:

(i) Single instruction stream - multiple data stream (SIMD).

(ii) Multiple instruction stream - multiple data stream (MIMD).

The terms "instruction stream" and "data stream" mean sequence of instructions and data respectively.

The SOLOMON (Simultaneous Operation Linked Ordinal Modular Network) system has been proposed for SIMD parallelism (106). In this architecture there is a central control unit executing the program commands. There are 1024 processing elements (PE) arranged in the form of an array. Each PE is connected to four of its physically adjacent neighbours. The control signals required to execute a command are given to the PE's by the central control unit through a network sequencer. Input-output from the P.E. network is accomplished by a 32 bit buffer register which has the capability to communicate with any specified row or column of processors. Each P.E. has two memory frames, each of 2K bits. Basically, each P.E. can be in one of the four possible modes and mode control is given by the central unit alongwith each instruction. If this matches with the P.E. mode state, then that command is executed. However, a P.E., not executing a command can provide an operand to one of its four neighbours. Its application in solving partial differential equation and photo reconnaissance in military applications have been explored (106).

The Illiac IV main structure consists of 256 PE's arranged in four reconfigurable SOLOMON type arrays (7). Each processor has 2K, 64 bit/word, 240 nsec thin film memory. It differs from SOLOMON in the sense that as much as four different instruction streams can be processed simultaneously and is suited for MIMD parallel processing. Its application in matrix operations, Fourier analysis and related problems have been discussed in (7). The four possible control units of Illiac IV improve the reliability of this system as compared to SOLOMON.

The distributed processor considered by Koczela (57) is a general purpose computer system with high reliability and is easily expandable or contractable. The organization consists of a number of identical cells interconnected in a certain manner. Each cell consists of a general purpose processor section and a small amount of memory (512 words, 16 bits/word) on a single LSI wafer. The cells are divided into groups and connected by intercell bus, and these groups are interconnected by intergroup communication switch. The difference between this architecture and SOLOMON or Illiac IV is that cells may be operated independently or dependently of the controller cell. Each group has a fixed number of cells, of which one will be designated as a controller cell. This facilitates the processing of a set of tasks or subtasks independently on distinct or the same data bases. This kind of parallelism is called "natural parallelism" by Koczela. Each cell has two neighbours and data can be shared between them. Similarly the source of instruction can be either from its own memory or from the controller cell in the group. Further, detection of faults, reconfiguration and its application in space mission have also been discussed in (42).

The multi list processor designed by Prywes is yet another architecture for information retrieval (84). The concepts of list processing and associative memory are used to process IPL like statements (77) with much greater efficiency. Representation of the information in the form of a multi-list for real time applications has also been studied by Prywes (85).

In the above architecture an associative memory is simulated on an addressed memory. It has been designed for variable length items. A memory synchronizer is placed between processor and memory and its functions are: (i) to store and read variable length items (ii) to automatically assign memory space and (iii) to synchronize the speed of the memory and the processor. The synchronizer and the memory together appear as an associative memory to the programmer.

In the above case, the particular formulation of information retrieval as a list processing problem is exploited through special hardware for matching variable length records. However, the parallelism is not exploited in this design. Though the memory synchronizer itself is a processor its function is to keep track of available free space and memory is addressed always through this synchronizer. However, it is claimed that the instructions for variable length data operations and the pseudo associative memory make programming much simpler.

The associative parallel processor (APP) proposed by Bird at Librascope group (15) is an architecture designed for picture processing. Associative memory is an integral part of this machine. The parallelism in problems will have to be made suitable to this architecture by "sequential-state-information" (31). When many pairs of data are to be added, in this system, they are added in parallel by word and serial by bit. Each cell in the associative memory has the required matching logic and other indicators. In addition to this associative memory, the computer has a random access primary memory and a processor to control the overall operation. The information stored in data registers are processed in the associative processor and results are gated out. Its application in picture processing has been considered in the reference given earlier.

In summary, the special purpose computer architectures for parallel processing have been proposed and designed in the literature. Array processing computer systems like Illiac IV with 256 P.E.'s each P.E. with 4K (64 b/w) memory and 10^4 gates are feasible with the current technology. However no parallel processing configuration for a computer system has yet been proposed for information retrieval applications.

2.4 Statement of the Problem

The process of Information Retrieval is achieved in three stages. These are:

(i) Pre-processing Stage: The query from the user is analysed. Synonym tables and file directory are consulted to determine the set of physical files to be searched for the given query.

(ii) Retrieval Stage: The selected files in the first stage are processed to find the set of pointers to the relevant documents for the query being processed.

(iii) Output Stage: The selected pointers in the second stage are used to pick the excerpts of the documents which can be any of abstract, the title, bibliographic details, the complete text. The selected excerpts are presented to the requester in appropriate form.

The first and the third stages involve interaction with slow user terminals. However, the retrieval stage receives its input, from the first stage and after processing them passes the selected pointers to the output processor. We observe that the first and last stages are I/O bound rather than processor bound. The case with the second level would very much depend on the I/O speed and processing rate. As the file used in this stage consists of indexed and coded documents, it is possible to store them on high speed back up memories like disks or drums. In a computer system designed for information retrieval applications it is worth exploiting this hierarchy of functions by considering well matched systems at these levels.

Consider an information storage with M items or documents and each document having N keys. Suppose, a request which specifies n attributes for the desired document is given. A document is said to be retrieved for this query if the corresponding attributes of the query and the document "match". Then it is not necessary for the retrieving machine to look at each document in a sequential order for match. In other words, if there are M such machines, they can examine the query and M documents simultaneously. Since the number of documents, in actual situation is large it is not economical to have that many machines. However, the documents can be classified into groups and these independent groups can be processed simultaneously. What we are looking for is a computer system with parallel processing capacity at the second level to exploit this "natural parallelism" (57).

As a different situation in retrieval operation consider the following type of query: "Does a patent on parallel processing computer for information retrieval exist?" Suppose there are M machines processing M different files to find the response to this query. If one of them comes up with an affirmative answer, the entire processing for this query can be suspended thereafter. Thus one processor in the second level might control the other processors working for the same query.

The SOLOMON architecture has parallel processing elements. The spatial adjacencies between the array of processors and the input output schemes are unsuited for the above problem.

In processing variable length records of information those processors which finish their job earlier will have to wait for others. Further each P.E., in this architecture is a general purpose computing element designed for operation on fixed length data. This makes it worse suited for information retrieval problems with variable length information items. Illiac IV being in the same lines as SOLOMON has the same disadvantages. These architectures have been suggested for problems in partial differential equations and weather forecasting. The distributed processor suggested by Koczela has the independent processing facility. But the grouping of cells and the lack of hierarchical processing make them less suited for retrieval operations.

The associative processors discussed in Section 2.3 have the same problem as the distributed processors. Further when the processing need matching of structured information the associative processor would be more complicated and also inflexible due to fixed logic. The multi-list processor by Prywes (84) is suitable for processing structured information but does not take into account the natural parallelism discussed earlier.

It can be shown that multiple attributes and user feedback in retrieval improve the performance of retrieval systems. The growing rate of information, need for interactive processing and multiple attributes all warrant a high computational power at a lower cost for information retrieval problems. The special characteristics of these problems can be exploited in a special purpose computer system.

A computer system for information retrieval should consider the following points:

(i) The natural parallelism in information retrieval problems.

(ii) The hierarchical structure of the problem.

(iii) Ability to schedule the available resources to the load on demand at any time, in order to provide better performance.

(iv) A modular structure which can be expanded or co~~u~~nt~~e~~racted to meet varying requirements.

(v) Reliability of operation of the system with the inherent redundancy in the system design.

However such an architecture has also its disadvantages. Two obvious disadvantages are the initial cost and the special purpose design. The trend in LSI (35) will reduce the cost in the long run. The growing trend to have mechanized information systems and information networks both at the national and international level would provide enough scope for such a specialized architecture.

In this thesis, a computer system well matched for information retrieval problems with due consideration given to the above points has been proposed. A system level simulation gives a methodology for system design of such an architecture. It is also possible to reduce the cost of such a system by making it highly special purpose as discussed in Chapter VI.

CHAPTER III

MULTI ATTRIBUTES AND USER FEEDBACK IN INFORMATION RETRIEVAL

3.1 Performance Improvement in Retrieval Systems

An item of information indexed and stored in a file for retrieval may have more than one "attribute". These attributes are also known as 'keys'. For instance, a document in a library information system may have the title, the author(s), the source of publication and the time of publication as its attributes. It has been shown by many experiments (93, 53, 69) that bibliographic information, citation and the journal of publication are useful attributes in retrieval. Such multi attribute representation of information can lead to higher precision when the requester knows his needs "precisely".

The need for precise statement of the query in terms of the language used by the system in describing the documents, is an important requirement. This precision is very hard to define commonly for all users. For example, an active research worker may be looking for a particular document whereas a beginner in that field may be interested in documents of general interest. This is an important consideration as the performance measures, namely, the recall and precision involve subjective decisions.

Thus it has been proposed by Salton (96) and others to involve the requester as part of the process of retrieval. Such online interactive retrieval systems seem to have advantages when indexing is not perfect. In this chapter these factors are discussed in detail and an algorithm is suggested for picking the attributes of the data base for an information system. The main objective is to emphasize the need for high computing power to be available to its users in an information system. The reasons for this need are two fold, namely, the multi attribute system of information representation and on line retrieval facility for interactive search.

3.2 Multi Attributes for Information Representation

In keyword indexing system each document is characterised by a set of keywords. The number of keywords depends upon the depth of indexing. A request in such a system also consists of one or more keywords. In many retrieval systems the user may assign weights to these keywords and give a threshold for selection. In simpler systems only conjunction or disjunction of these keywords are permitted. Conjunction of keywords or co-occurance improves precision. For instance, the request for documents on computer system design is more precise when the keywords "computer" "systems" and "design" are required to co-occur, in a document for retrieval. On the other hand disjunction of these keywords might retrieve more unwanted documents. However, if a document, d_j deals with some aspects of computer design and if the three keywords mentioned above are not co-occuring, the conjunction method would miss it whereas the use of disjunction would retrieve it. This difficulty arises mainly because the concerned document is not viewed by the indexer as it would be done by this particular requester. Such mismatches will be henceforth referred to as "imperfect indexing". It is particularly impossible to achieve perfect indexing as the requirement differs from individual to individual.

The main difficulty in mere keyword oriented search stems from the problem of synonyms. The two terms "computers" and "calculating machines" may be synonymous in some context but not always. For a requester interested in modern computer systems design, document dealing with early calculating machine design is irrelevant. Identification of the validity of this synonym, independent of the user is difficult. Under such conditions a second attribute to the document, say, time of publication would be useful. Either the apriori knowledge of the user regarding this attribute or interaction with the system which might throw light on this will reduce the irrelevant documents retrieved.

Extending this idea of multi attributes, one is interested in knowing what are the attributes of the documents that should be used in the file organization and query formulation. As in any design, there is a trade off between the number of attributes and their cost of inclusion. The inverse relation between recall and precision is shown in Figure 3.1, both for

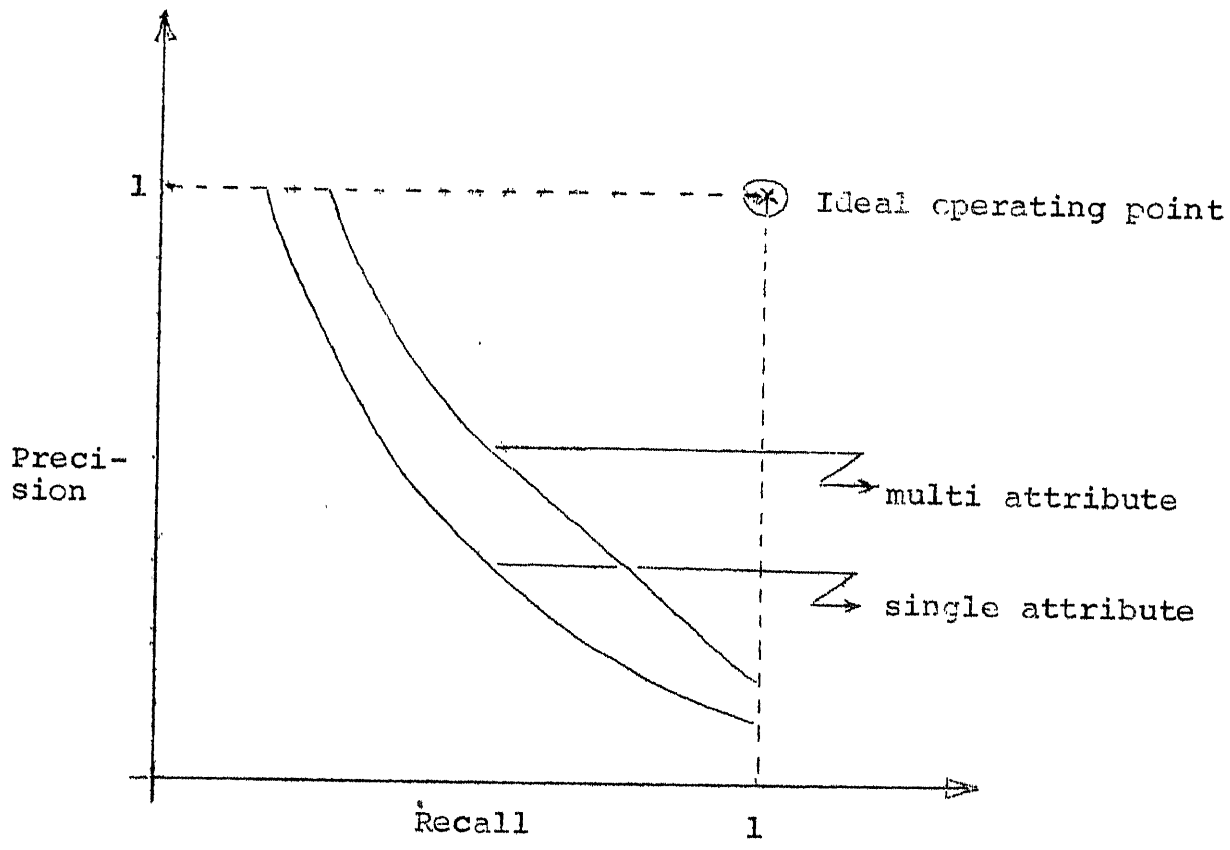


Figure 3.1: Performance curves for single and multiple attributes (see reference 69).

single and multi attribute systems. It should be observed that the performance under multi attribute system has improved where the ideal performance is unity precision and unity recall.

Let the documents be represented with 'n' attributes in the data base. Consider the following definitions:

Definition 1: The document space or data base denoted as D is defined as the collection of N documents where each document is characterized by an n-vector.

Each component of this n-vector describes one attribute. For example, a set of keywords or index terms and a set of citations could be the two attributes in a particular system.

Definition 2: Request is defined as a non-null n-vector possibly with null elements. The elements of this vector have the same interpretation as in definition 1.

Definition 3: Answer to a request R denoted as D* is defined as

$$D^* = \{d_j \mid f\,[M(R,d_j)]\} \quad \text{for all } d_j \in D$$

where M is a relatedness measure function and

f is a decision function or predicate which determines,

if a document is to be retrieved or not for the given reques

In a weighted term search, the relatedness measure is obtained as the sum of the weights of the matching terms. If this sum is greater than the threshold τ, it is retrieved. The threshold τ is normally given by the requester or is determined by the system on an interactive basis.

Definition 4: An indexing scheme is said to be perfect if the precision of retrieval is unity.

By selecting one of the attributes from a n-attribute system, it is possible to derive n different single attribute systems.

Theorem 1: A multi attribute system of information representation and retrieval is at least as good an approximation to perfect indexing as the single attribute system derivable from it.

Proof: An irrelevant retrieval of a document occurs due to one of the following reasons:

(i) inaccurate description of the request, that is, what is described in the query is different from what is wanted.

(ii) imperfect indexing.

Let $P_j(d_i)$ be the probability, that the document d_i is retrieved as relevant due to matching of attribute j alone but later found irrelevant, possibly due to one of the reasons mentioned above. Then $P(d_i)$, the probability of document d_i being retrieved relevant with n attributes matching and later found irrelevant would be

$$P(d_i) = \prod_{j=1}^{n} P_j(d_i) \leq P_k(d_i) \qquad 1 \leq k \leq n$$

$1-P(d_i)$ will be a direct measure of precision. A multi attribute system does no better to a query when the equality holds.

The intuitive idea of improving the precision using more attributes is stated by this theorem. In a practical system one would be interested in overall performance of the system for a representative set of queries. This could be increased by the choice of proper set of attributes and also educating the user whenever possible regarding the attributes not mentioned by him. (See Figure 3.1). The inverse relation between recall and precision would result in the loss of recall when precision in increased. Retrieval is a trade off between these two and on line control is possible for them in an interactive systems.

<u>Theorem 2:</u> For small changes in request space, change in answer set D* with multiple attributes is less than or equal to that with single attribute under unity precision and recall.

<u>Proof:</u> Let D_r^* be the answer set for the request r. Assuming ideal operating conditions, namely, unity recall and unity precision, any small variation in r, say Δr should not add any new document to D_r^* nor delete anything from it. Consider the variation in query (request) due to imperfect indexing and synonym associations.

$$r = \langle r_1, r_2, \ldots, r_n \rangle$$

$$r + \Delta r = \langle r_1 + \Delta r_1, r_2 + \Delta r_2 \ldots r_n + \Delta r_n \rangle$$

The probability of adding a new document to D_r^* in a single attribute system, say with the first attribute is due to $r_1+\Delta r_1$ matching with that of a document not in D_r^*. In a multi attribute system this will be due to combined simultaneous matching of the n attributes and this joint probability would be less than or equal to that due to one attribute. Similarly one can show that the probability of deleting a document from D_r^* is less in multi attribute system and hence the theorem.

However if the variation in request is in only one attribute then the multi attribute system does no better than the corresponding single attribute system. A well selected set of attributes and user interaction would reduce such cases and statistically improve the performance with multiple attributes.

3.3 A Case Study with Library Information System

The bibliographic information of documents has been used in retrieval. For example, Salton (93) has suggested a scheme to pick keywords for describing a document not only from the text but also from the cited documents. The citation index or bibliographic coupling used by Kessler (53) ignores the keyword information and bases citation as the only tool for retrieval. Both these methods are not complete when considered individually. A multi attribute system of information organizatior for a library information system is suggested as follows:

Every document in the index file is characterized by the following five attributes:

1. Author or creator of the document
2. Time of publication or creation
3. Name of the document or title

4. Index terms or keywords for description

5. Citations or subject affinity.

The attributes of this system, shortly called ATNIC, are chosen from their common usage in manual systems. Different indexing schemes currently in use form proper subsets of this multi attribute system.

(a) Author indexing	<A T N Ø Ø>
(b) KWIC indexing	<Ø T N I Ø>
(c) Citation indexing	<Ø T N Ø C>
(d) Subject indexing	<Ø T N I Ø>

Each indexing has its own merits and demerits. For instance, citation indexing is technically simpler but essentially keeps track of the information in the past. Also, a user should know a representative document to formulate the query. The combination of these indexing schemes as a five attribute system combines the advantages. When the user describes his needs precisely with more than one attribute, it is possible to improve precision. Suitable feedback from the system and query reformulation might improve also the recall.

Now, it is possible to define a relatedness measure for ATNIC system using the last two attributes. Consider the following definitions.

Definition 5: For a collection of m documents the ij^{th} element in "document-document association matrix", $[B]_{mxm}$

indicates the association between ith document and j-th document.

$$B_{ij} = B_{ji} \qquad \text{symmetric measure}$$
$$B_{ii} \overset{\Delta}{=} 0 \qquad \text{by definition}$$

This association measure may be obtained from the common key-words, common citations etc (53).

<u>Definition 6:</u> For a collection of n index terms a term-term association matrix, $[T]_{nxn}$ is defined such that T_{ij} is the association between the i-th and j-th index terms.

Association between two index terms may be obtained from their statistical co-occurrence. This matrix essentially defines the "closeness" between various index terms (96).

<u>Definition 7:</u> For a collection of m documents with n index terms a discriminant matrix $[A]_{mxn}$ is such that the ij-th element is a measure of the association between the i-th document and j-th term.

This association may be obtained by considering the term - frequency in the document (96).

<u>Definition 8:</u> A modified discreminant matrix $[D]_{mxn}$ is defined as follows:

$$[D]_{mxn} = [A]_{mxn} \times [T]_{nxn}$$

The resulting matrix is similar to A, but takes into account the synonym between terms in the document corpus.

Each document is now characterized by the corresponding row vector in D matrix. Similarity between two documents d_1 and

d_2 can be defined as

$$\text{sim}\ (d_1, d_2) = \sum_{i=1}^{n} \frac{1}{|d_{1i} \cdot d_{2i}|} \quad \text{for } d_{1i} \neq d_{2i}$$

Though the above measure does not satisfy the metric properties given below, it is possible to define distance measures with such property (Appendix I).

$$d(A,B) \geq 0 \text{ and equal to 0 implies } A = B$$

$$d(A,B) = d(B,A)$$

$$d(A,B) + d(B,C) \geq d(A,C)$$

Definition 9: Relatedness between documents is the reciprocal of the distance between them.

Definition 10: Among three documents A,B and C if relatedness of A to C is obtained implicitely through another document B, then A is said to be related to C indirectly.

Theorem 3: An indirect relatedness can atmost be equal to the direct relatedness between two documents.

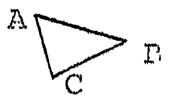


Figure 3.2

Proof: Let the relatedness between two documents A & B be R(A,B); then the theorem states

$$R(A,B) \geq R(A,C) + R(C,B)$$

$$\frac{1}{R(A,B)} \leq \frac{1}{R(A,C) + R(C,B)}$$

$$\frac{1}{R(A,B)} \leq \frac{1}{R(A,C)} + \frac{1}{R(C,B)}$$

$$d(A,B) \leq d(A,C) + d(C,B)$$

(which follows from the above definition and metric property). Hence the theorem.

The theorem explains the intuitive idea that when relatedness between documents is established through a third document, there may be unwanted retrieval. However, when the three documents are strongly related, that is the equality holds in the above relation, it does not introduce any spurious response.

This argument can be extended to show that this indirect relatedness is less effective as the level of indirectness increases. An indirect relatedness is said to be n-level if it goes through n distinct documents.

Corollary: An n-level indirect relatedness can be at most equal to (n-1) level relatedness. Proof of this follows directly from the definition and is given in Appendix II.

A similar approach to define a metric using 'C' attribute in ATNIC becomes difficult (6). Consider the following relatedness measures.

Definition 11: A citation matrix $[C]_{nxn}$ for a collection of n documents is a binary matrix whose ij-th element is 1 if i-th document cites j-th document or else it is zero.

Then relatedness between two documents, say i and j

$$R_{ij} = C_{ij} + C_{ji} + \sum_{k=1}^{n} C_{ik} C_{kj}$$

This gives the number of common citations between i and j and the number of times they are cited together. This measure is symmetric, non-negative but the triangle inequality is not always satisfied. But in perfect indexing scheme we observe this measure will be a metric. For example, consider the following condition (See Figure 3.2).

A cites B and C.

C cites B and no two of them are cited together.

Then $R_{AB} = 1$

$R_{AC} = 1$

$R_{CB} = 1$

Hence $R_{AB} \leq R_{AC} + R_{CB}$.

In perfect indexing scheme when two documents are related they will always be cited together. However, in practice the citation index refers to the documents in the past and would also depend on the availability of documents to the writers.

In multi-attribute systems like ATNIC, a composite measure of relatedness considering all the attributes is required. Relatedness with regard to other attributes in ATNIC, namely author, title or time of publication is but a simple match condition. Relevance of a document to the given query can be obtained as the weighted sum of the relatedness of different attributes used for search.

3.4 Attributes Selection for File Organization

The desirable characteristics of the attributes in a multi attribute system would be:

1. Independence: The attributes should be as independent as possible so that maximum information is conveyed jointly.
2. Use in context: The significant keywords in the title are also in the index terms attribute. However, the title when presented in its complete form is more useful, in particular for user feedback.
3. Usage: The attributes should be the ones often used. This factor justifies their inclusion in the data base.
4. Compatibility: Compatibility with existing indexing systems is desirable for facilitating file organization and data collection.

The five attributes suggested in Section 3.3 satisfy most of these requirements.

The following algorithm is proposed for the selection of attributes for a data base from a set of m possible attributes.

Definition 12: Self information of an attribute is defined as log q_i where $q_i = 1 - p_i$ and p_i is the probability of the i-th attribute being used in a query.

From this definition it is clear that one may not have such data when the file is organized for the first time. However, such data can be collected for systems in operation to aid other systems design as well as for self evaluation.

Definition 13: Joint information between two attributes is defined as $-\log p_{ij}$ where p_{ij} is the joint probability of the two attributes i and j in the data base.

When the joint probability is unity, the above measure becomes zero and these attributes are same, as far as retrieval is concerned. When p_{ij} is zero this measure is maximum and the two attributes do not co-occur at all.

Let a joint information matrix for a collection of n documents be $[I]_{nxn}$ whose ij-th element is the joint information between i-th and j-th attributes. Then, a method to select n attributes for the data base is as follows:

Step 1: Enumerate all the possibly relevant 'm' attributes of the data base. This m is greater than n, the number of attributes that one wants to use in file organization.

Step 2: Obtain the self information of attributes and the joint information matrix I. This might need sampling of the data base when it is large.

Step 3: Choose as the first attribute the one having the maximum self information.

Step 4: To choose the next attribute look along the row (or column) of the I matrix pertaining to this attribute. The attribute having maximum joint information with this is chosen.

Step 5: To choose the third and subsequent attributes one can adopt the same rule with the following modification. Each attribute which is already chosen can be assigned a weight based on its self information. Note this self information is a measure of the probability of that attribute being used for query. The weighted sum of the row (or column) vectors pertaining to the already chosen attributes gives a single vector. The situation now is identical to step 4. If the number of attributes selected is less than n, then go to step 4.

In an information system design, file organization, query language, and retrieval method are highly inter-related. In a query language design one would like to improve the recall by the union of attributes which will account for insufficient knowledge of the user about his needs and also any variation in indexing. Similarly precision can be improved by conjunction of attributes. However, it must be remembered that they are inversely related. Consider a query as follows:

$$q = S_1, S_2, S_3, \ldots, S_n$$

where S_i's are such that their joint information considered pair-wise is minimum. This would improve precision. If each S_i is a union of n_i attributes one can account for combined advantages of both. Such a scheme has its own disadvantages as increasing the number of attributes would increase the storage and retrieval time.

Using the above algorithm and the five attributes mentioned in Section 3.3 a case study was conducted with a sample data*. As the name or title attribute is unique and different for different documents in the sample chosen, it is not included in the I matrix. Let us consider a hypothetical case wherein each document has one author and one index term only. Then P_{AI}, the joint probability

* Articles published in Comm. of ACM. Jan-Dec. 1969.

of the author and information content attribute is given by

$$P_{A_1 I_1} = \frac{\text{No. of documents with } A_1 \text{ and } I_1}{\text{Total No. in the sample}}$$

Then

$$P_{AI} = \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \, P_{A_i I_j}$$

for m authors and n index terms.

This summation is to be normalized to lie in the range (0,1).

When there are k index terms for a document, the k dimensional vector is converted to a scaler by assigning weights to these keywords. Probability indexing mentioned earlier (Section 2.2) is useful in assigning these weights. P_{AI} obtained for a pair of documents is averaged over the sample space to obtain a single entry in I matrix. One can define other joint probabilities following the same technique and the resulting I matrix is shown in Table 3.1 where the unit of information is "nats"

Table 3.1

	A	T	I	C
A	0	3.835	3.726	4.029
T	3.835	0	3.849	4.001
I	3.726	3.849	0	3.992
C	4.029	4.001	3.992	0

I-Matrix for the case study

From previous experience index terms attribute is chosen as the attribute having maximum self information and the rest are assigned equal probability. This leads to the following ordering

I,C,T and A. The experiment concludes that if an information system is to be designed with two attributes only, I and C are recommended for this data base. In any practical system it is desirable to keep track of the frequency of usage of the attributes in the system as well as those not in it. This would be useful in evaluating the selection strategy and modify it when required.

Having considered some results and aspects of multi-attributes in information retrieval, the following sections describe the use and effect of user feedback in improving the performance of retrieval systems.

3.5 A Model for Retrieval Systems

An information retrieval system is a collection of indexed documents, requests for search and one or more retrieval methods. The retrieval method associates with each document a relevance number for a given query. From this point of view it seems appropriate to use the notion of fuzzy sets (114) in defining a retrieval system.

Definition 14: Fuzzy Set: Let Y be a space of points whose generic element is Y such that Y = {y}. A fuzzy set A in Y is characterized by a "membership function" f(y). For every member of Y the function f(.) assigns a real number in the interval [0,1], f(y) is called the grade of membership of y.

Definition 15: Retrieval Process: A retrieval process is an ordered triplet <D, $f_R(.)$, R> where D is the collection of indexed documents, $f_R(.)$ is the membership function defining a fuzzy set D* in D and

R is the given request.

In a multi attribute system using n attributes the request will be a n-dimensional vector and D will be a set of n dimensional vectors. The membership function is the algorithm for the relatedness measure between a document and the request. Consider the following definitions and the theorem due to Salton (96).

Definition 16: An inclusive information retrieval system may be defined as a partially ordered information retrieval system, such that for all $r, s \in R$

$$r \geq s \Rightarrow T(r) \subseteq T(s)$$

where R: set of requests

$\geq$ a partial order relation in R

T(·r): set of documents retrieved for the query r.

Theorem 4: In an inclusive information retrieval system, the image under T of a decreasing chain in R is an increasing chain in 2^D (set of all subsets of the documents) and the image of any increasing chain in R is a decreasing chain in 2^D.

This theorem states the common observation that more specific the request - or equivalently, the larger the number of terms used to formulate the request - the smaller is the size of the retrieved document set. According to this definition a standard keyword system is inclusive. The same notion can be defined using the fuzzy set model as given below, which is not restricted to keyword matching.

Definition 17: Inclusive information retrieval is a class of requests whose membership functions are identical.

Two membership functions $f_A(.)$ and $f_B(.)$ defined over the set X are identical iff

$$f_A(x) = f_B(x) \ \forall \ x \in X$$

This definition is not restricted to the keyword system. A more specific and more generic request is, now represented by a threshold. A document is retrieved as a response to the given query if its grade of membership is greater than or equal to this threshold. A larger value for this threshold means more specific retrieval and vice versa.

Definition 18: A relation between two requests $r, s \in R$ is defined as

$$r \geq s \quad \text{iff} \quad \tau_r \geq \tau_s$$

where τ_r and τ_s are the corresponding thresholds associated with r and s. The relation $\geq$ is reflexive, antisymmetric and transitive When the membership functions are same the request with larger threshold retrieves lesser number of documents and vice versa. This explains the above theorem due to Salton.

This formalism, has the advantage of being simpler to interpret. Notions of probabilistic indexing can easily be made part of the relatedness measure or the membership function.

3.6 User Feedback in Retrieval Systems

There are two possible ways of improving the performance of an information system; namely, multiple attributes and user feedback. During an online search procedure, information about the retrieved documents are displayed to the user. This creates an interaction with the user to reformulate the query when needed. The display may be excerpts of stored dictionaries term list, term frequency information, related index* terms, titles or abstracts of some selected papers. The information about the retrieved and displayed document being relevant or irrelevant could be used in modifying the query. Results obtained from one such experiment with SMART system is shown in Figure 3.3. It has been found in that experiment that the information used from the retrieved irrelevant documents in formulating the query is quite useful.

Though the relative merit of the information obtained from relevant or irrelevant documents cannot be categorically stated, the use of feedback is found to improve the performance in general.

Definition 19: A feedback retrieval process is defined as an iterative process and the i-th stage in the iteration is characterized by

$$<D^i, \ f_r^i(.), \ R^i>$$

In any practical information system we can assume the document collection D and the membership function $f_r(.)$ do not change

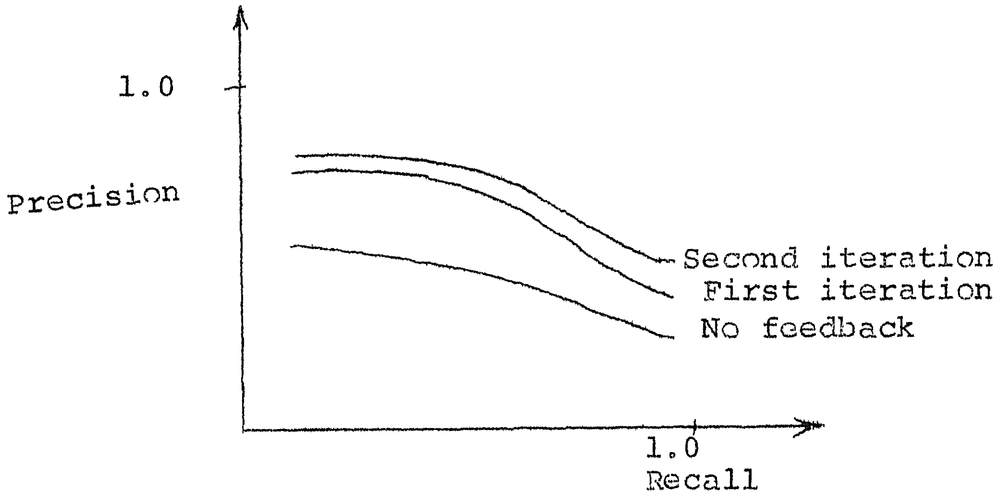


Figure 3.3(a): Feedback with relevant documents

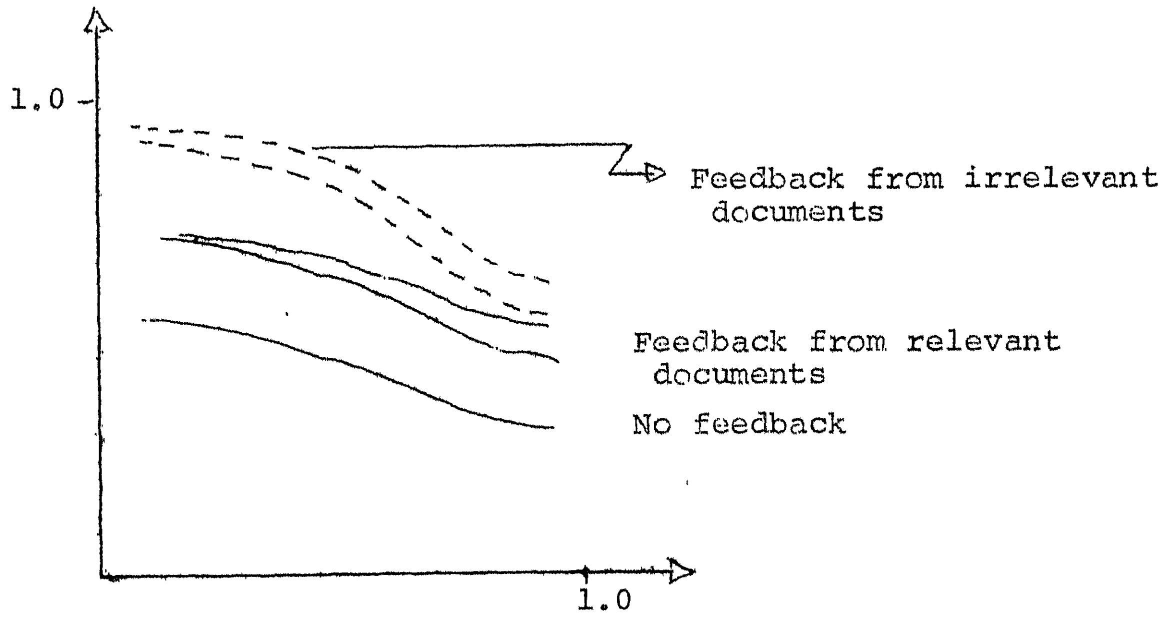


Figure 3.3(b): Feedback retrieval performance (for two iterations) (see Ref. 18).

for the given query. Thus the modification in the process is essentially in query formulation. However for systems like on line ticket reservation it is not uncommon to consider the modification in D also.

For a given query let the answer set from an ideal system be D*. Then D* will contain all relevant documents in D and all documents in D* will be relevant. Though the precision is somewhat easy to measure from the user response it is difficult to measure the recall. One way to do this in an interactive system would be to make a more generic request and see whether any new relevant document is being obtained. However one has to consider the point of diminishing return in this iterative process.

Let D_r be the approximation to D*. As D* is not known apriori, a possible way to decide the termination of the iterative process would be to check if

$$D_r^{i} = D_r^{i-1} = D_r^{i-2} \ldots D_r^{i-n} \quad i \geq n$$

The equality may be considered not only from the relevant and irrelevant documents point of view but also with the cost and value of the modified answer set.

The i-th stage modification of the request would depend on the previous stage and its answer set. Similarly the threshold for selection will be modified.

$$R^{i} = G(D_r^{i-1}, R^{i-1})$$
$$\tau^{i} = H(D_r^{i-1}, \tau^{i-1})$$

G and H are called "modifying functions". In general, guidelines can be given to form more generic or more specific queries for a particular indexing scheme. But the modifying functions G and H will depend very much on the requester and his relevance and specificity judgements. However, it is felt that a statistical study of the users in such feedback system may be helpful in classifying the users into equivalence classes and characterize these functions in some sense. As a modification to the G and H functions one can consider the use of past information beyond one previous stage in modifying the request at the ith stage.

Definition 20: A retrieval process S_1 is said to be better than another retrieval process S_2 iff there exists ~~a~~ R for which $M(D_{r1},R)$ is greaterthan $M(D_{r2},R)$ where D_{r1} and D_{r2} are the answer sets obtained for the query R from S_1 and S_2 and M is an evaluation function.

In its simplest form the evaluation function can be weighted sum of the grade of memberships of the elements of D_r. The weights assigned may be positive or negative according to the judgement of the requester.

Theorem 5: A feedback retrieval process is better than the corresponding non-feedback process.

The validity of this theorem can be proved by the following argument. Let D_r be the answer set obtained for the query R with threshold τ. It is always possible to obtain R' and τ' which would drop out atleast one relevant document or add an irrelevant document to D_r. If one starts the retrieval process with R' and τ' the answer set obtained would be D_r'. But by construction

$$M(D_r, R) < M(d_r, R)$$

It is possible to obtain R from R' by properly defining the modifying functions. Suppose the indexing is perfect, then it is possible to include proper attribute or to modify the threshold in an interactive system to get back the dropped out document.

However, feedback might give increased irrelevant documents when it is not used carefully. It will be the responsibility of the user to keep track of the sources of such increased irrelevant retrieval and avoid them by careful query formulation in subsequent iterations. The experiments with SMART system and project TIP have shown that on an average 20-30% improvement in recall and precision is possible with feedback retrieval (181).

It is quite legitimate to think at this stage that the multi attribute system being a better approximation to perfect indexing should be able to do better in feedback retrieval. The probability of the user interacting with the system is also more in multi attribute system. This intuitive idea is said in the following theorem.

<u>Theorem 6</u>: An n attribute feedback retrieval process is at least as good as (n-1) attribute process with feedback.

Following similar arguments as used in multi-attribute system with probabilities and the existence of at least one request for which it does better as in the above theorem, the proof of this theorem is direct.

Consider a keyword system with user feedback and a system with keywords and citations or bibliographic data with user feedback. The theorem states that the latter will perform better for at least one request. The experiments have shown in keyword system that feedback improves the performance on a statistical average; but it remains to explore the performance of a multi attribute system and the choice of attributes for an on-line interactive system.

In this chapter some results on multi attributes and user feedback in information retrieval have been discussed and earlier results obtained in the literature are restated. The main motivation is to indicate the growing need for specialised computational power in automatic information retrieval. In this context, it is worthwhile to exploit the special nature of the information retrieval problems and design a special purpose computer system, well matched for this area of application. With growing needs in diverse areas of information retrieval such system design will be of general interest to a widening class of users. In the succeeding chapters we will explore this specific problem in detail.

CHAPTER IV

COMPUTER SYSTEM ARCHITECTURE

4.1 Hierarchy of Operations in a Retrieval Process

The three levels of an information retrieval process discussed earlier, namely, query pre-processing, retrieval and dissemination have different requirements. For instance, the "matching" required in the first level is not as much as that in the second level. In a typical case the first level processor may analyse the query and determine the physical files to be searched. There may be a synonym dictionary and a file directory to be consulted in determining these physical files. As the system will be used in interactive mode the first level processor will have the facility for time sharing.

The requirements of the second level processors are different. The communication with slow user terminals are taken care of by the first and third level processors. The main function of the second level processors is to select the set of pointers to the relevant documents for the given query. This would involve the input from the index file, internal processing to select the relevant documents and communicating the selected pointers to the third level processor. As the selected files for search can be examined in parallel there may be more than one second level processor in the system. In such cases the number of processors allotted to one query, at any time will depend on the number of available free processors and computational requirement of the queries awaiting service. Unlike general purpose time sharing system (28) it is possible to estimate the computational requirement more precisely for each query. The sum of the expected processing times of the files selected for search, in response to a given query will give this value.

The third level processor has comparatively less matching and computation to perform. The pointers selected by the second level processors are used to pick their corresponding texts or abstracts from an abstract file. The selected excerpts are then presented to the user. For an interactive system it would be preferable to have a soft copy display. The complexity of this third level processor might vary depending on the allowable cost. For example, in its simplest form the pointers may constitute the "call number" of a selected document or the bibliographic reference to the selected documents and they can be directly presented to the user. The hard copy facility may provide the abstracts of the selected documents on request. It is possible even to store the entire text on computer operated microfilms (COM) (2) and provide the user with a hard copy of the selected document. In the simplest electronic catalogue system, operating in an university environment the soft copy output of the pointers to the selected documents might be sufficient.

Considering the requirements of these three level processing, a computer system with hierarchy of processors is proposed. In this system there can be more than one second level processor operating concurrently. The first and the third level processors being limited in their functions, it is possible to consider them as special purpose time sharing systems. Such an approach would minimize the total cost of the system. The overall organization of the proposed computer system is shown in Figure 4.1. The inter processor control and data communication among the second level processors is possible only under the control of the first level processor. When such inter processor activity is low as in information retrieval problems, the overhead will not be prohibitive. The unibus shown in Figure 4.1 constitutes the control and information path between the first level processor and the second level processors.

4.2 Requirements of a Computer System for Information Retrieval

The need for interactive processing as discussed in Chapter III for information retrieval process requires the first level processor to be a time sharing system. The time sharing is the simultaneous employment of a computer by several users and each one has the illusion of being the only person using the machine. The main hardware and software features of a time-

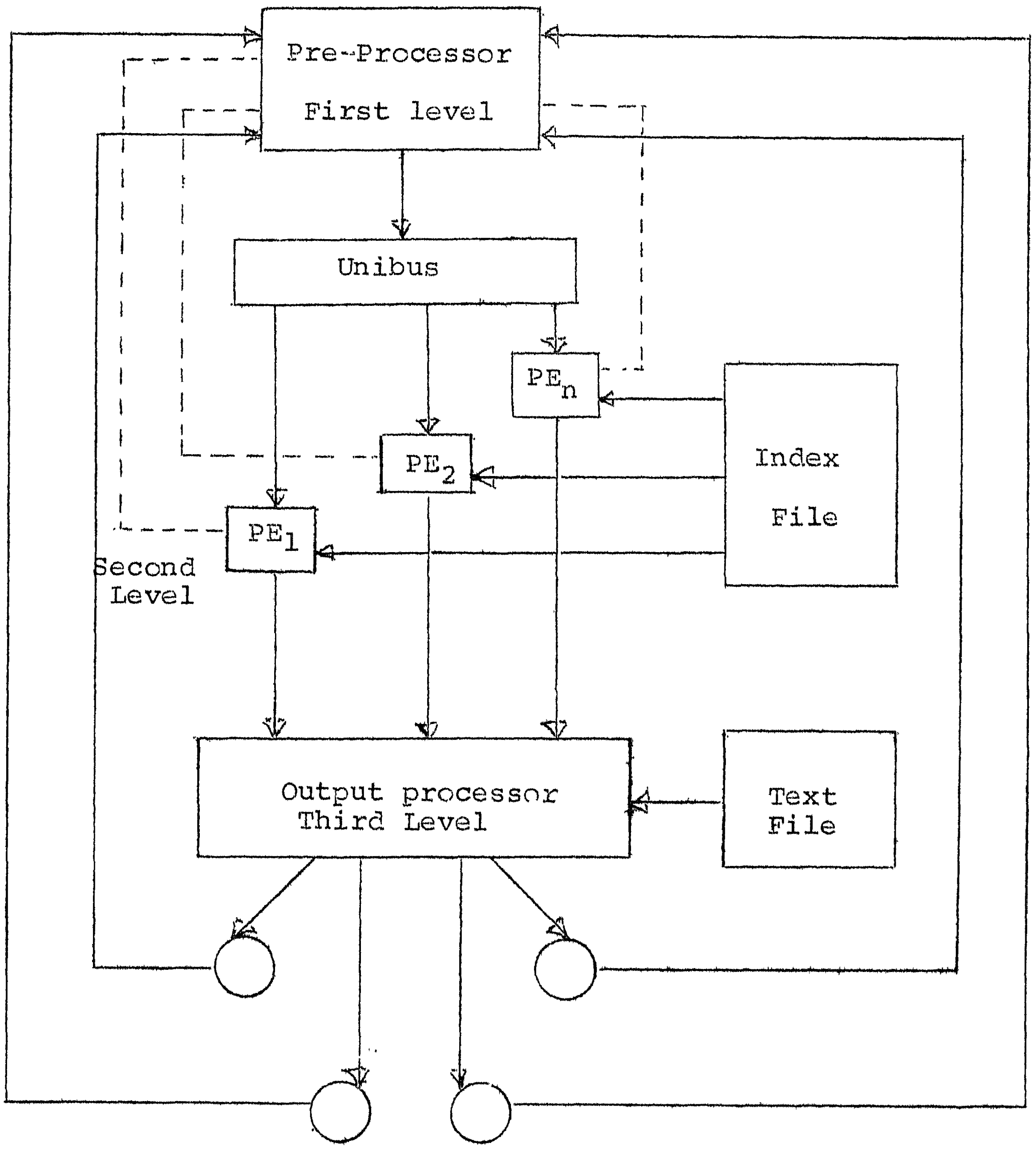


Figure 4.1: Computer system with a hierarchy of processors.

sharing computer are as follows (82):

Essential hardware features:

(a) Memory Protection

(b) Interrupt feature

(c) Built in clock

(d) Dynamic program relocation

(e) Auxiliary Storage

(f) Typewriter and display consoles.

Essential software features:

(a) A supervisory program for memory allocation and management in general

(b) A command language for user communication

(c) A debugging language for on line debugging

(d) A symbolic programming language to simplify program writing.

The above requirements of a time-sharing system have been considered for general purpose computing in which problems of different kind are solved by users (100). A special purpose time sharing system for a particular application would need much less effort in the operating system development. For instance, in the time sharing system of the proposed architecture one would be restricted to problem oriented query language. Developing a processor for a query language for information retrieval would be much simpler compared to compiler type languages like FORTRAN or ALGOL. For these query languages the debugging system would also be simpler. In such a system the users at the console will be using a problem oriented query language and symbolic programming will be reserved for the system implementation and modification.

However, the hardware requirements mentioned earlier are necessary even for the special purpose time sharing systems. The decreasing cost of hardware with integrated circuits have shown that the major part of such systems would be governed by the cost of the software (55). and hence the total cost of a special purpose time sharing system would be less. The complexity and the functions of the third level processor are much different. The pointers selected by the second level processors are used to pick the corresponding abstracts and get them displayed at the appropriate console. In other words this processor will be executing a fixed program all the time. For example, this processor may have a COM file and a fixed program to do the following:

(i) To collect all the pointers selected for a particular query.

(ii) To pick the appropriate abstract from the abstract file.

(iii) Logic required to switch from one user terminal to the other.

The new product announced in (115) has a disk memory and can serve more than thirty of its terminals each equipped with a keyboard and a CRT display. This, with a little modification can function as an output processor. When one needs to have flexible priority scheme in servicing the terminals this output processor would also become a programmable time sharing system. A fixed priority terminal service and fixed program execution seems to be adequate for most of the information system requirements. However, it should be observed that it is possible to completely avoid the output processor and endow this power to the second level processors. For example, in simplest "electronic catalogue" systems where only accession numbers or the bibliographic details of selected documents are presented, it is possible to make the second level processors to do this display function. But the communication with the slow output terminal would degrade the performance as the fast second level processors would then be waiting for the completion of the output function. In this architecture such communication with the slower terminals are separated from the second level processors.

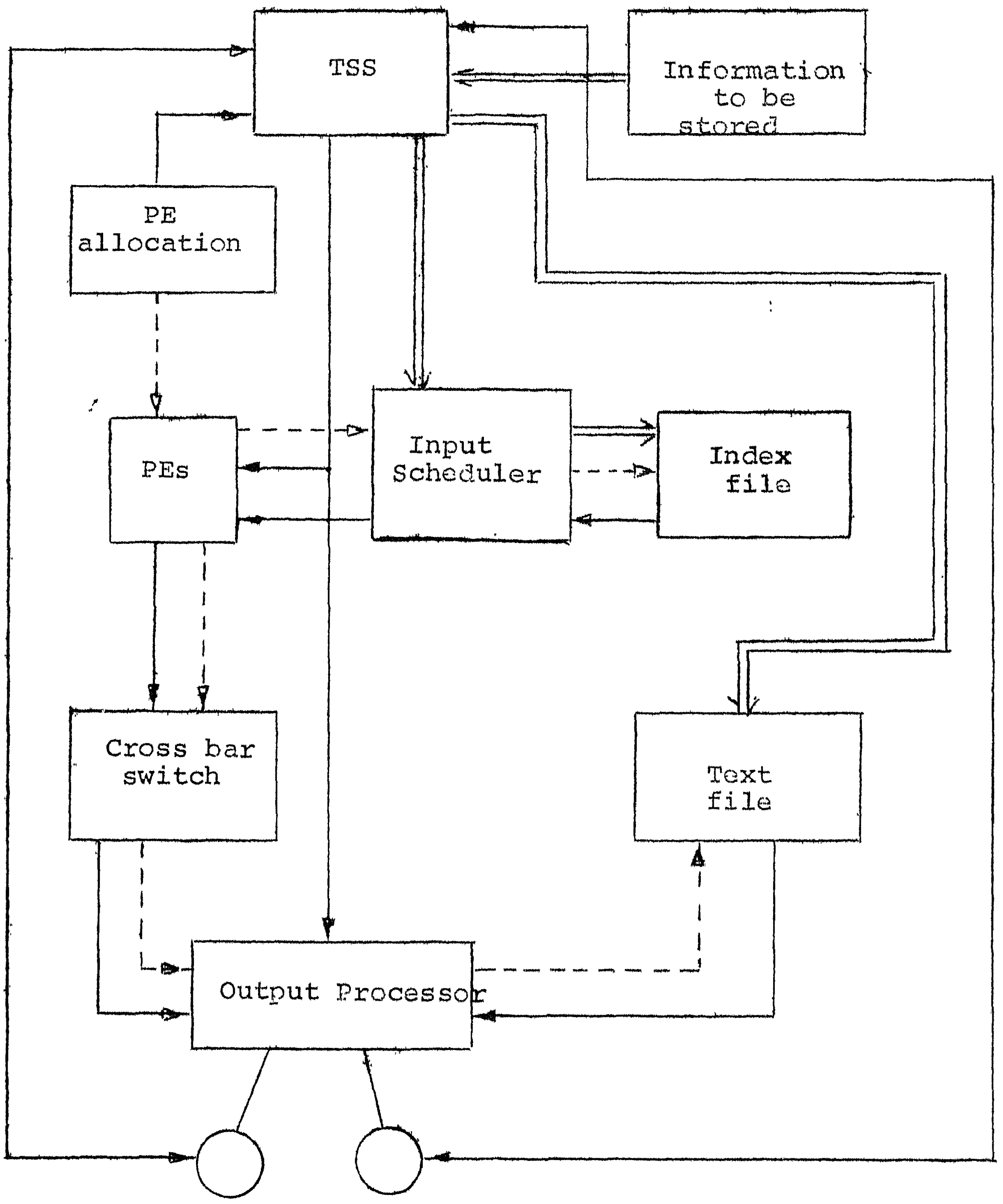
4.3 Architecture of the Proposed System

The flow of information between various functional blocks in the system are as shown in Figure 4.2. Essentially there are two phases:

(1) Information organization and storage

(2) Information retrieval.

These two phases of operations could be mutually exclusive for a library information system. In other words, during the short period of file updating query processing will be suspended.



—— Information path

- - - Control Signal path

Figure 4.2: Control and information flow in the hierarchical system.

An user logs in from any one of the free terminals as in any time sharing system. The central TSS (Time Sharing System) responds to the user request after analysing his pass word. The query processor residing in the core of the TSS is called for service when the query from the terminal is completely received. This program in association with the dictionary of synonyms and the file directory determines the set of physical files to be searched for the given query. The file directory would depend on the file organization and the nature of the physical file. For example, in a serial file organization and tape file system the directory would give the number of initial files on the tape to be skipped to get the physical file needed. During the retrieval phase the TSS is executing the same fixed program with different data which arise from the terminals. Thus it is possible to code this program in reentrant fashion (43).

The information generation phase is the one during which information items are added to the files. The flow of information in this phase is shown with double lines in Figure 4.2. During this phase, the second level processors could possibly be used for automatic information analysis.

After determining the set of physical files to be searched for a query, the TSS allocates a certain number of free processors to this job. Unlike computations with a general purpose computer, a quick estimate of the processing time is possible. The processing algorithm and the size of the physical file are fixed. This combined with the number of keys in the query will give an estimate of processing time required per query. Based on this, the scheduling routine allocates one or more processors to the job as available.

The TSS has a list of pointers to the processors waiting for a task. When an assigned processor completes itstask, it interrupts the TSS and this free processor is added to the above list. When this list is empty the jobs wait for processors. The TSS, sends control signals to the selected processors. These control signals establish the connection between the TSS and a second level processor for data communication. The attributes of the query alongwith the files to be searched, are then passed to the P.E. (Processing Element) and stored in their memory. The P.E.'s become autonomous thereafter in completing their functions. The given query is used in searching the assigned physical files. The set of all selected pointers are sent to the output processor memory.

The output processor would have a common back up storage accessible to the P.E.'s (through their output channels) as well as to the output processor. On this back up storage each terminal will have an assigned user area for storing the selected pointers. When there is one output channel for each processor receiving information in parallel, the cross bar switch shown in Figure 4.2 is not necessary. However, when the number of processors is large and the simultaneous transfer of pointer information is less probable, a small number of output channels will suffice. The cross bar switch then provides connection between any of the output channels and a second level processor. The output processor might do some processing on these selected pointers. For instance, it might give the count of the number of documents retrieved or might present the documents of higher relevance first.

In this architecture, that part of the processing which needs user interaction and deals with slower terminal devices are separated from the rest. The PE's have comparatively less time spent in input and output. The internal processing by the PE's for retrieval could be simple keyword matching or matching structured information like graphs (23) which form the two commonly used data types. Considering the flexibility in implementing various algorithms with these data types the PE's are made programmable. However, the memory to store these programs and the instruction set will be much smaller. Some aspects of the design of a typical processing element are discussed in the next section.

In summary, the hierarchical computer system has the following major subsystems:

(i) A time sharing system to process the requests for retrieval and a software query processor.

(ii) A processor allocation program

(iii) An unibus used for communication between P.E.s and the TSS.

(iv) A set of dedicated processing elements capable of concurrent operation.

(v) A cross bar switch to connect any of the PE's to one of the output channels.

(vi) An output processor to receive the pointers selected and display the information.

(vii) An abstract file and an index file.

(viii) A set of user terminals with a keyboard and display or typewriter whichever is feasible.

(ix) Update programs for information storage files.

The TSS and output processor with human interaction for time shared operation would pose no new problems (71). On the other hand restricting the design of these two time shared systems to their special task would considerably reduce the cost of the total system. However, the considerations in the P.E. design would be different and are discussed in the rest of this chapter.

4.4 Some Considerations in Computer System Design

Any design is an iterative process and strikes a trade off between the needs of the user, called system architecture and the needs of the fabricator, called system engineering (19). The system architecture of a P.E. is discussed in the following which includes engineering considerations, so that the design will be economical.

Mode of Operation: The P.E. has the following modes of operation:

(i) Run complete mode

(ii) Run interrupt mode

(iii) Program mode.

In the first mode the P.E. completes its function by completely processing the assigned physical files. At the end of this operation it interrupts the first level processor and joins the list of free processors. In the second mode, when a response is found for the given request it automatically interrupts the TSS after communicating the selected pointer to the output processor. The interrupt service routine in the TSS adds this processor alongwith the other processors alloted to this terminal to the list of free processors and the query processing is complete. This mode of operation is useful in "patent search" and so on. In program mode, the assembled program to be executed by the P.E. is loaded into the program memory of the selected P.E. from the first level processor. When the number of such programs are not many it is possible to think of fixed programs stored in magnetic cards. In addition to this external interrupts, the P.E. has I/O interrupt facility. The I/O interrupt scheme is much simpler than a general purpose system as the I/O devices are fixed and less in number. The input to the P.E. comes from the index file and the output is to the output processor.

<u>Memory requirement</u>: There exists a definite speed mismatch between the faster CPU of the P.E.s and the slower back up memory in which the index file is stored. A primary memory of proper size would alleviate this speed mismatch. The two conflicting requirements in the selection of the memory size for the P.E. are the cost and I/O boundedness. When the processing time of one core load of the memory is smaller as

compared to the access time of that information from the back up file, the P.E. is said to be bounded by input. Similarly one could define the output boundedness. However the possibility for output boundedness is small as the information transferred is low. The preliminary memory design would need an estimate of the average processing time per query per document, average access time of the file and the storage required per document. A modular organization of memory (19) in banks would be useful not only in accomodating later expansions but also in using the memory banks for simultaneous processing and I/O transfer (33). It is to be noted that the memory size selection should be to exploit the full processing power of the P.E. by avoiding the I/O bounded condition. Memory size required, using the data obtained from a sample experiment is discussed in the next section.

C.P.U. Requirement: Architectural design of the CPU is intended to be oriented to end-use and proceeds backwards. Some factors to be considered in this design are in sequel: (26)

1. Data type formats
2. Word size
3. Address spaces
4. Registers and uses
5. Minimum operations set
6. Required address modes
7. Instruction set
8. Instruction format.

Data types: Data types commonly encountered include: integers, real numbers, individual characters, English text, symbol strings, pictures and directed graphs. Basically the data in information retrieval problems are variable length records. Often various coding schemes are used to code the descriptors into a fixed length code (87). The number of such code words per document is variable in number. The common data types used in these problems are integers, characters and graphs.

Word Size: All the third generation computers have word organized structure with the facility for character handling. As the first iteration the lower bound on the word size is obtained by considering the range of data values. The upper bound on the number of bits per word is by the economic constraints and average utilization of the memory. In this case the P.E.s are dealing with indexed and coded files and there is no need to have extended character set. A word size of 18 bits gives a sufficient integer range for storing the pointers in one word for a file as large as 256 thousand documents. The recent trend in LSI memories (81) shows that the cost of memories vary linearly with word length. Also two words (36 bits) when used jointly for a keyword provides sufficient range for coding the various keys in the system. The use of such word length is common in some low cost mini computers (103). However, a 16 bits or 12 bits word length with multiple word length format also would satisfy most of the requirements. Since the keywords are stored as fixed length codes for ease of processing, the packing density of the memory will not be affected when it is a sub-multiple of the code word length. On the other hand a larger word length memory reduces the number of memory accesses. When the word length is too long, however, packing of code words will be necessary for effective utilization of the memory.

Address Spaces: In this part of system design distinct address spaces or areas of storage are defined and the data types mapped into them. Unlike the general purpose computers the P.E.s operate with restricted type of programs which are changed only under the control of the system. In the limit, these programs can be wired in the processors as discussed in Chapter VI. Hence there is no need for a large address space for programs. However, programmability endows some flexibility when they are used in different information systems.

Registers: The recent trend in computer system design is toward a multiplicity of general purpose registers rather than a separate index, arithmetic, I/O and extension registers (11). However, the complexity of the control unit design with such general purpose registers is much more (88). The use of multiple registers in the C.P.U. in matching operation and handling variable length data would be more useful and they are discussed in Section 4.5.

Minimum Operations: The basic set of minimum operations on the data types are summarized here. This ultimately will evolve into an instruction set. The P.E.s need to have the following operations: Integer arithmetic, matching, and Boolean operations.

Address Modes: The same argument of restricted type of program execution and dedicatedness of the P.E.s leads to the choice of direct addressing. Indexing, base address displacement, indirect addressing, addressing through registers, are all nice, but are not always necessary (26).

Instruction set and format: Defining even a first-cut instruction set and format is an elaborate art. This process is crucial to computer architecture. As a means to modify or to incorporate new instructions, microprogramming is suggested in the literature (48). However, microprogramming has its own disadvantages. Considering the main operations involved and the factors discussed earlier the following types of instructions are proposed in the first round:

(1) Load registers from memory

(2) Store in memory

(3) Integer Add

(4) Skip on conditions

(5) Match

(6) Branch

(7) Subroutine branch

The above factors discussed in designing a computer system in general are reconsidered for the P.E. design in the next section using some special notations.

4.5 PMS and ISP Description of the Architecture

A computer system is complex in several ways. There are at least four levels in which the complex system can be described

1. PMS (processor, memory, switches) level
2. Programming level (Instruction-Set processor level)
3. Logic design level
4. Circuit level.

Each level arises from the abstraction of the levels below it. The PMS and ISP are the two descriptive notations developed by Bell and Newell (12) to describe the computer system in the top two levels. In this, the informal notations in use for describing the computer system are made consistent and more powerful. More than thirty computer systems have been described in this notation by Bell and Newell (13) and is chosen here for its compactness.

The BNF description and semantic interpretation of these formalized notations are given in the book cited earlier and Appendix III gives a brief introduction to the notation. The various components of PMS level description are given the notations as shown below:

1. Computer (C)
2. Processor (P)
3. Data (D)
4. Transducer (T)
5. Control (k)

6. Switch (S)
7. Memory (M)
8. Link (L).

In ISP level, notations have been developed to describe the following:

1. Data types
2. Instructions
3. Operations
4. Processors

Description of a computer system in PMS level is the detailed study of the memory, processor and various switches in the system and their characteristics. We will, now consider the memory size requirement of a typical P.E. in the proposed architecture.

Let

s = the average number of instructions executed per second by the P.E.

n = the number of words in the P.E. memory

m = average number of words required per query

k = mean access time of the index file

t = transmission time per word to memory.

Then

$$\frac{n*m}{s} = k + nt$$

would make the processing time equal to the input time. When there are two memory banks each with n words, they could be used in the same way as swinging buffers are used in I/O channels. This in turn avoids input bounded conditions.

Number of words in memory, $n = \frac{sk}{(m - st)}$

For the value of s as 500,000; m as 10; k as 30 milliseconds for a typical disk monitor system (11) and t as 2 micro seconds, the memory size required would be about 2K words. In the PMS description of a P.E. two banks of memory with each P.E. will be assumed. The cost of such a memory size does not seem to be prohibitive as the cost of LSI memory is decreasing (98).

The PMS description of the complete architecture is shown in Figure 4.3 at the structure level and expanded in Figure 4.4. However, the components description of the first and third level processors are not shown. The number of simultaneous terminals to be handled by these time sharing systems would depend upon the average arrival rate and the average time the user stays in the system. The PMS descriptions of such time sharing systems and their design methods are discussed in literature (11, 13).

The PMS diagram of a typical second level processor is shown in Figure 4.4(b). It consists of a processor whose ISP description is given later. The data storages M_A and M_B are two memory banks. The switch S_2 connects either of them to the index file for information transfer. The control, K_1 connects the processor to the common data bus which forms the

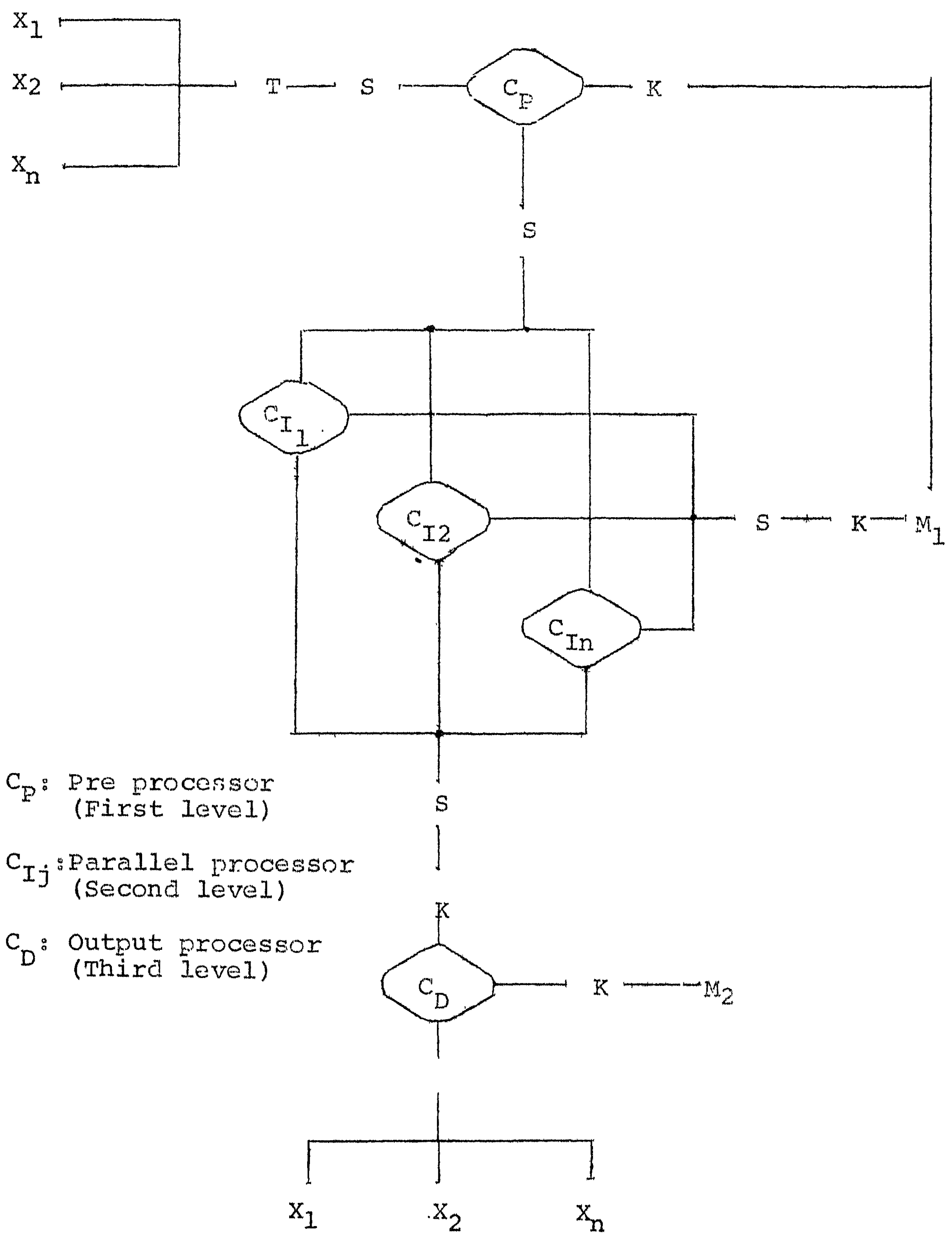
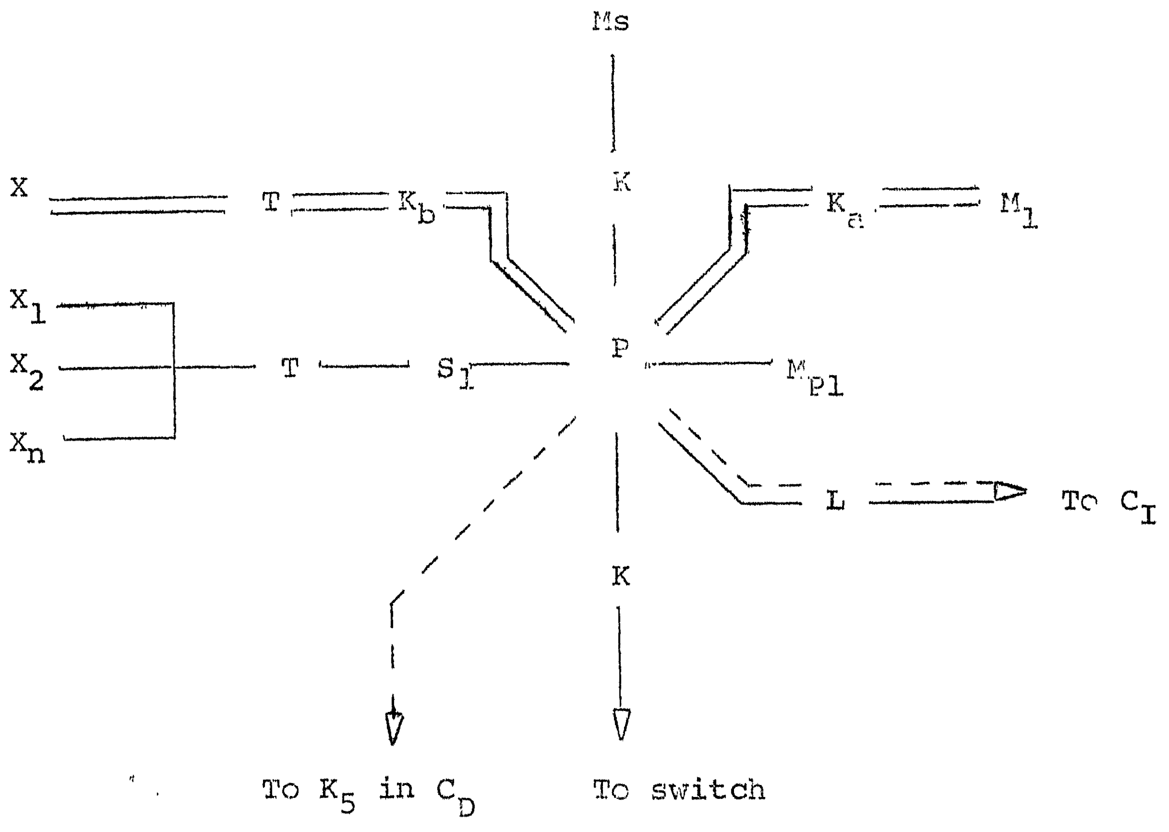

X_1
X_2
X_n
T
S
C_P
K
S
C_{I_1}
C_{I2}
C_{In}
S
K
M_1
C_P: Pre processor (First level)
C_{Ij}: Parallel processor (Second level)
C_D: Output processor (Third level)
S
K
C_D
K
M_2
X_1
X_2
X_n

Figure 4.3: PMS description of the total system



——— Information signal path for retrieval

– – – – Control signal path for retrieval

═══ control and information path for updating the index file.

Figure 4.4(a): PMS diagram for first level processor (C_P).

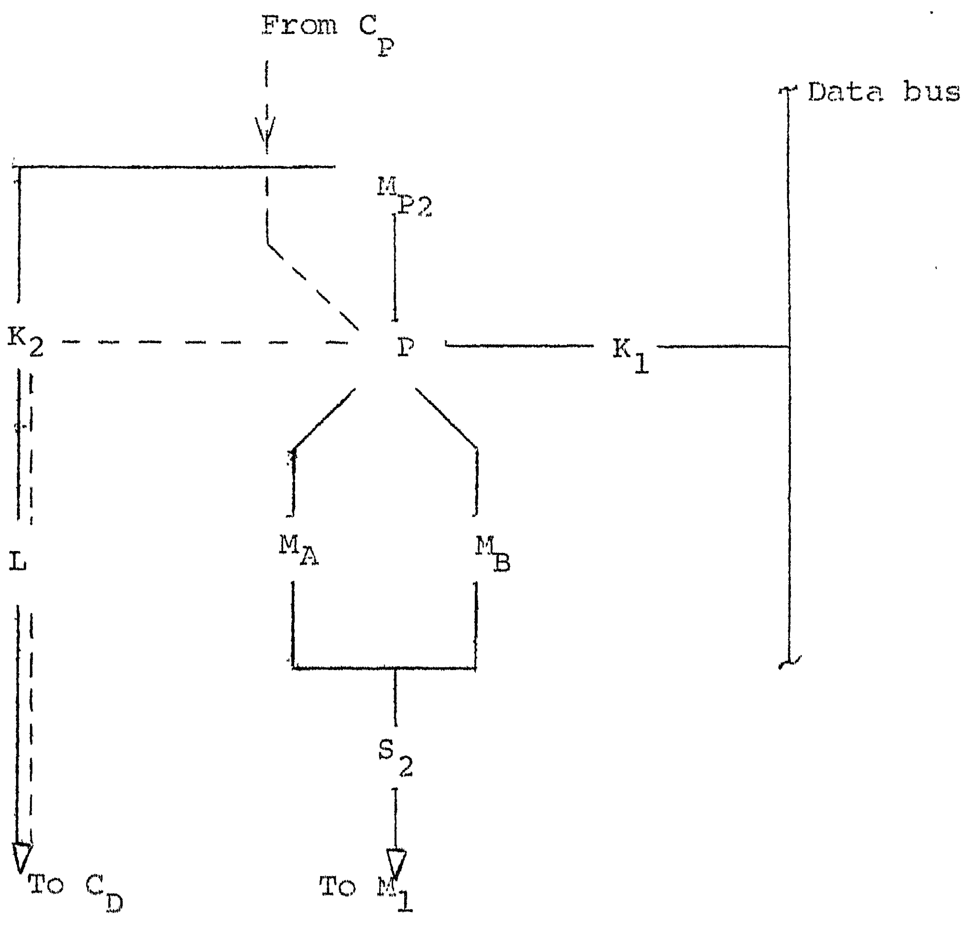


Figure 4.4(b): PMS diagram for second level processor (C_I)

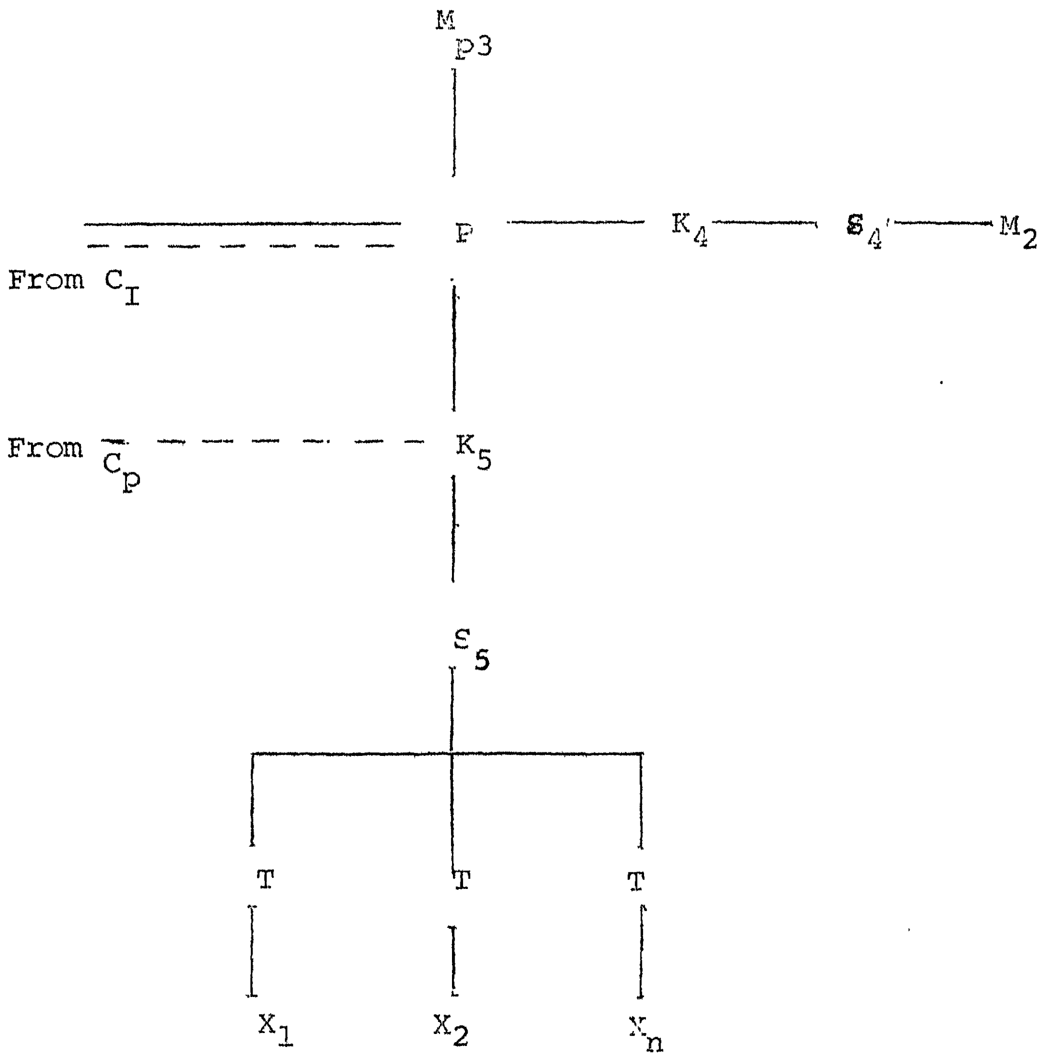


Figure 4.4(c): PMS diagram for third level processor (C_D).

central broad-cast line for information transfer from the first level processor. For instance, each P.E. can be associated with a unique identification code. On receiving this identification code the appropriate P.E. is connected for further information transfer through the data bus. The control K_2 and link L establish the connection between the P.E. and the output processor. All P.E.s allotted for a single task will be connected to the same output channel. This connection between the P.E.s and output channels is established by a cross bar like switch S_5 under the control of K_5 and is directly controlled by C_p (Figure 4.5).

The M_{p2} constitutes the primary memory for the P.E. and is used for storing the programs for execution. These programs can be assembled in the first level processor and stored in these memories at the beginning. A memory with 1K words would be sufficient for storing the fixed programs and the query attributes. When the number of different programs executed by the P.E. is not many, they can be stored in fixed memories (72) and query attributes might be stored in M_A or M_B.

As the index registers would add to the hardware cost it is avoided. However, the indirect addressing feature is cheaper to implement (26) and also useful in setting up loops in programs and is thus included in the ISP description.

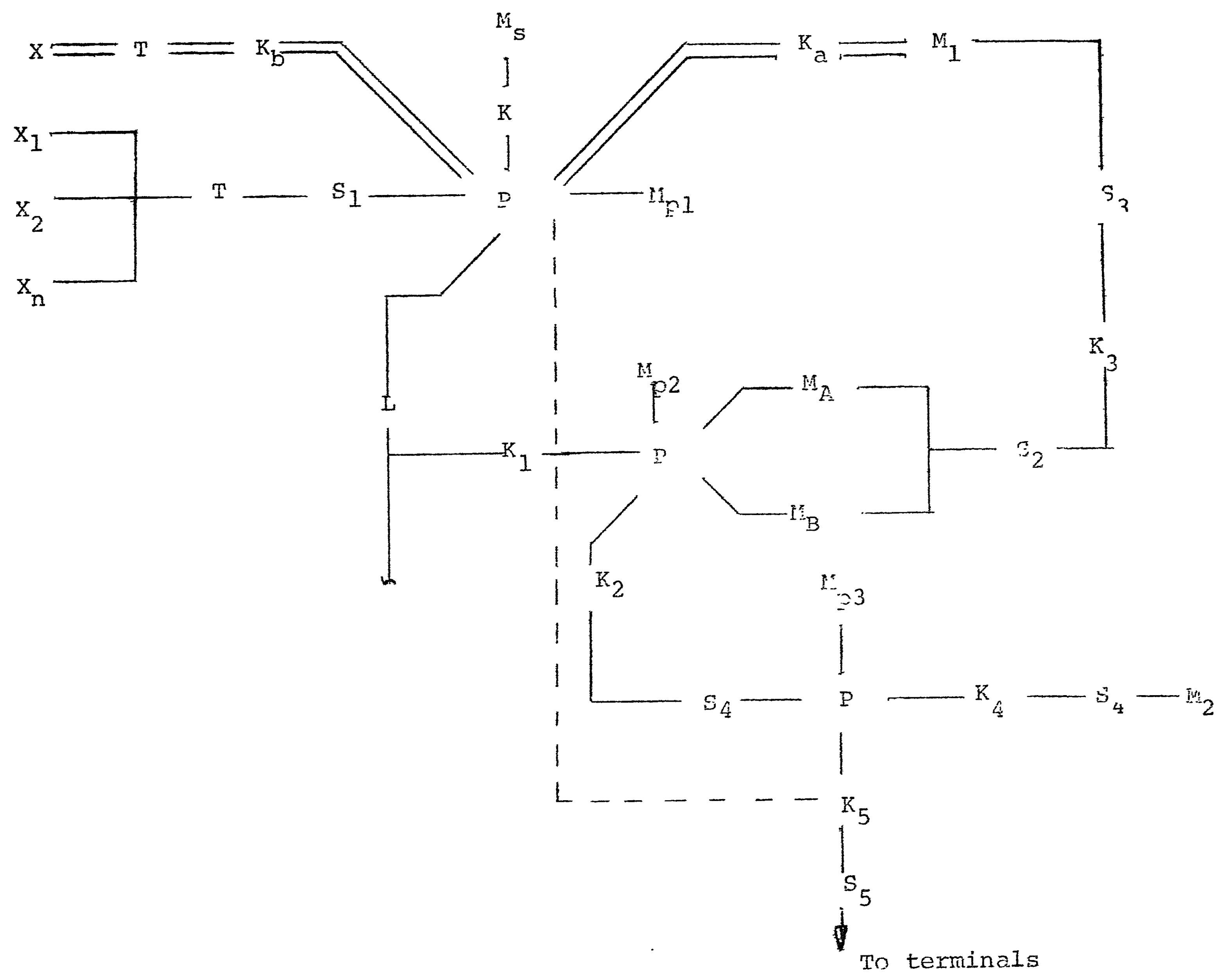


Figure 4.5: Combined PMS diagram with one second level processor

ISP Description of a P.E.*

Processor State:

A <0:17>	Accumulator register
L	One bit extension for accumulator for overflow and carry.
R	One bit indicator. It is ON when the program is executed.
Interrupt disable	1. when interrupt is to be disabled. Program controlled.
Interrupt type 1/IT1	1 when the P.E. is interrupting the C_p.
Interrupt type 2/IT2	1 when the P.E. is interrupted for input from M_1.
Interrupt type 3/IT3	1 when the P.E. is interrupted for output.
Interrupt type 4/IT4	1 when the P.E. has to interrupt the C_p.
Program Mode/PM	1 when the P.E. is in program mode and controlled by C_p.
M	Mode selector to select one of the modes of operation discussed earlier.
PC <0:13>	Program counter
BI <0:1> = PC <0:1>	Memory bank indicator

Memory State:

M_p $[0:1777_8]$ <0:18>	Memory bank 0
M_A $[0:3777_8]$ <0:18>	Memory bank 1
M_B $[40000_8:77777_8]$<0:18>	Memory bank 2

* 18 bits/word has been used for the ISP description.

Instruction Format:

```
     Instruction/i            <0:17 >
      OP   <0:5 >              := i<0:5>
      Indirect bit/ib          := i<6>
      address/addr <0:10 >     := i <7:17 >
```

Address Calculation:

```
      Z <0:13>   := ( Effective address

                          ¬ ib → Z" ;  Direct addressing

                            ib → M(Z"))  Indirect addressing
       Z"  := BI □ addr               Memory bank and address
                                      selections.
```

Instruction Interpretation:

```
               R ∧  ¬I    →     (instruction  ←M[PC];
                                 PC ← PC + 1; next
                                 instruction execution)
               I →   (M(0) ← PC ; PC ← 1   interrupt disable ← 1)
```

Instruction set and Instruction Execution Process:

The following is the tentative instruction set. Data types and the operations are given prime consideration in selecting this set and no simulation at the function level has been done.

```
     get → (A ← M[Z]);       Clear and add
     Sto → (M[Z] ← A);       Store
     tra → (PC ← Z);         Transfer
     tmi → (sgn→  (PC ← Z));   Transfer on minus
     Sgn    :=   A <0>
```

sbr $\rightarrow$(M[Z] $\leftarrow$ PC; next PC $\leftarrow$ z+1)	subroutine jump
Sts $\rightarrow$ (M[Z] $\leftarrow$ M[z]-1);	Subtract one from storage
skm $\rightarrow$ (Z' $\rightarrow$ PC $\leftarrow$ PC+2);	Skip on storage minus
Z $\rightarrow$ PC $\leftarrow$ PC+1)	
Z' := M[z] <0>	Sign bit of the storage addressed
add $\rightarrow$ (A $\leftarrow$ A + M(z))	addition
sub $\rightarrow$ (A $\leftarrow$ A - M(z))	subtraction
skE $\rightarrow$ similar to skm but	
Z' := A $\otimes$ M[z]	skip on equal
skZ $\rightarrow$ similar to skm; but	
Z' := (A=0)	skip on zero
and $\rightarrow$ (A $\rightarrow$ A $\wedge$ M[z])	AND operation
lbi $\rightarrow$ (BI $\leftarrow$ i <16,17>)	Load base register for memory bank selection.
sfl A $\uparrow$ 2^n	Shift left n times
sfr A $\uparrow$ 2^n	Shift right n times

The instruction set given here is not complete by itself but could evolve as a final set with further function level simulation of the P.E.s.

Consider the following set of multiple registers and "associative" matching as part of the P.E. which would be useful in keyword matching. Suppose the processor consists of two sets of registers as shown in Figure 4.6 and denote them as R and D. The R registers hold the query descriptors. If the number of registers in the set R is insufficient to

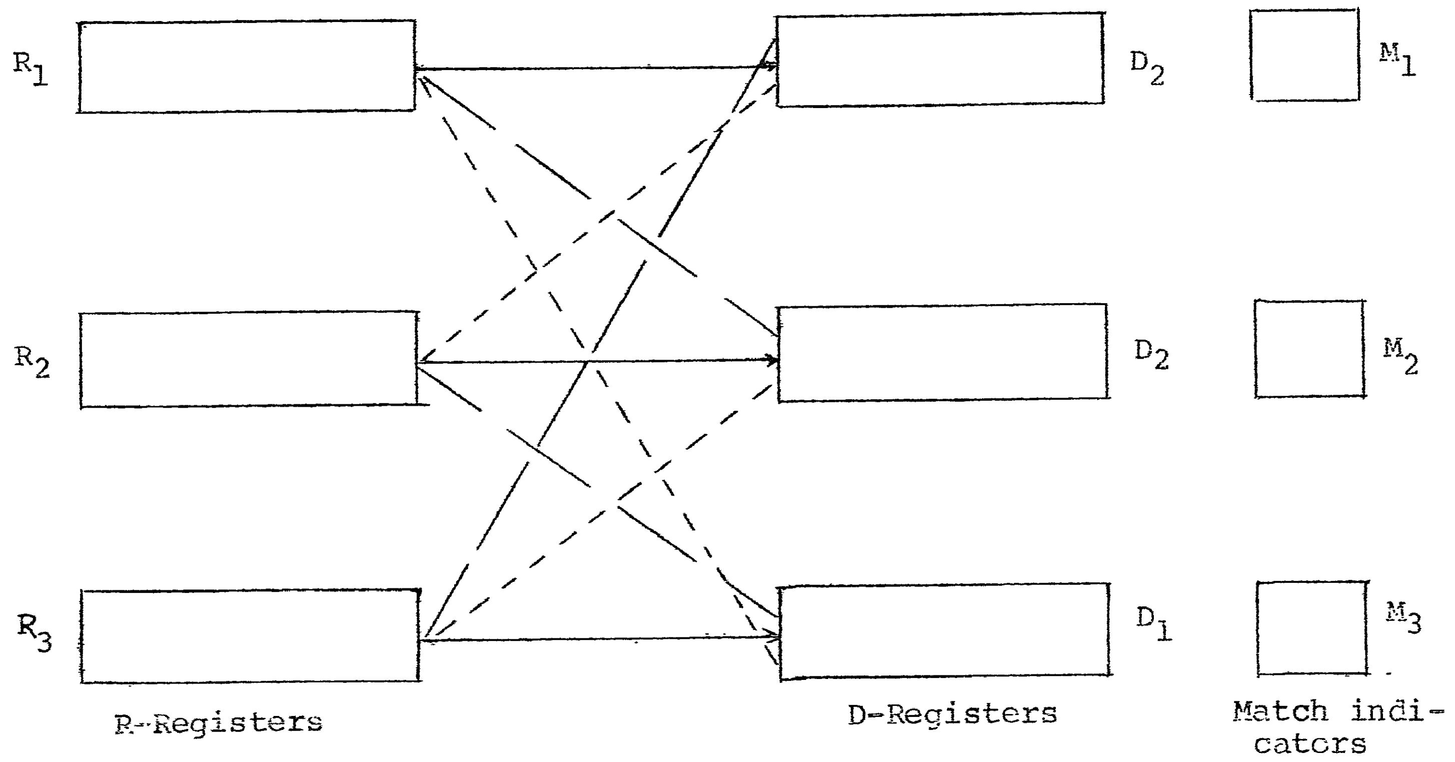


Figure 4.6: "Associative" match registers

hold all the descriptors of the query, only part of it will be in R registers. Similarly the D registers hold the descriptors or the attributes of a document. Then it is possible to envisage the following instructions for matching.

When a match parallel instruction (MPL) is executed R_i register is matched with D_i bitwise and if the match is successful the corresponding match bit (one per R_i) is turned ON. The match parallel complete (MPLC) instruction does similar matching for all possible pairs. For example, when there are three registers in R and D sets, in the first cycle of this instruction $(R_1; D_1)$, $(R_2;D_2)$ and $(R_3;D_3)$ will be matched. In the second and third cycles the following pairs are matched.

Second cycle:	$(R_1\ D_3)$	$(R_2\ D_1)$	$(R_3\ D_2)$
Third cycle:	$(R_1\ D_2)$	$(R_2\ D_3)$	$(R_3\ D_1)$

At the end of this instruction execution the match bit of a request descriptor is ON if it matches with any one of the document descriptors in the D register. This will be useful in keyword matching and achieves the effect of an associative memory in a limited sense.

The above parallel match scheme can be used with the multiple load instruction. The "get multiple" or multiple load (GETM) instruction loads the R or D registers from a list having variable number of descriptors stored in the P.E. memory. When the list is complete which indicates the end of descriptors in the query list or document list, a corresponding indicator is turned on. Conditional skip instructions which test these indicators will be useful in this context to handle variable length data. An instruction for testing the match bits of the R registers and to accumulate selected weights would realize the weighted term search much efficiently. This "associative" match combined with the rotary file discussed in Section 5.6 will be suitable for variable length data handling.

4.6 About the Architecture

In the above architecture any interprocessor communication among the second level processors is under the control of the first level processor. The need for such communication can be minimized by judicial file organization. For instance, consider the ATNIC system described in Chapter III. One way to organize the files is to use inverted file organization for each attribute and let different processors search for different attributes. This would need heavy interprocessor communication. On the other hand a sequential file organization or combined file organization, where all the attributes of a document are handled together by one processor would avoid such inter processor communication. However, the search for a particular item as in the case of patent search would need inter processor control. When one of the processors assigned for the job finds the required document, all the processors assigned to this job are to be suspended from further processing for this job. This inter processor control is accomplished through the first level processor.

An important advantage of the architecture proposed and the autonomous nature of second level processor is its adaptability. A P.E. with one particular type of design can work alongwith another P.E. designed totally in a different way. For instance, the use of associative memory may be made part of the new P.E.s added to the system and the structure of this new P.E.s will not affect the functioning of the other P.E.s.

Another advantage of the modularity in this system is the reliability obtained through equipment reconfiguration. When a P.E. fails, it can be automatically excluded from the system for servicing. This results in degradation in service but the system is fail safe. This feature will be valuable as large nationwide information systems and information networks become popular.

CHAPTER V

SIMULATION OF THE PROPOSED ARCHITECTURE

5.1 Some Problems in Simulation

Simulation and analytical models constitute the methodology, in general, in analysis, design, and evaluation of systems. The behavioral characteristics of a system for different values of the parameters which characterize it can be studied through these models without actually building the system. The increasing trend in using digital computers for simulation of various deterministic and stochastic systems have led to the development of many simulation languages (20). The simulation language GPSS (44), developed for queueing system simulation and CSL (107) for continuous system simulation are few examples.

Often simulation is a time consuming task. Whenever the simulated system is structurally very much different from the simulating system, this problem is severe. For example, the simulation of a multiprocessor system on an uni-processor computer poses a number of problems (49). The objective of the simulation is always to be kept in mind in designing a simulation model. A function level simulation with all registers and counters of a multiprocessing system will be much more involved than a system level simulation which is an abstraction of the function level behavior.

Analytical models developed using probability theory queuing theory (28) and Markov processes (21) have some use in the analysis of computer systems. However, the analytical models are approximations to physical systems and are limited by the techniques available for solving them. It is possible to simulate a system to a larger detail at the cost of simulation time on a computer. But, it is not always possible to model a system to any finer level and analytically solve them. Analytical models are less time consuming and favourable in that sense. The results obtained in a closed form are much easier to manipulate and use.

In the simulation of multi-processors the following difficulties are common (49):

(i) Degradation of ability to differentiate between times of event occurrences.

(ii) Simulation of simultaneously occurring events on a sequential machine.

(iii) Development of a language for describing the jobs to be processed by the multi-processor system.

In most of the event-oriented simulation, a monotonically increasing master clock is used. Event times are calculated during the course of the simulation and added to the master clock time (T_m). As time increases, the ability to distinguish between the times of event occurrences decreases. Hence, it should be observed that the event time added to master clock is not too small. (For a binary computer with n bit mantissa this should not be less than $T_m/2^n$).

The programs to be run on the simulated multi-processor system are to be described in an appropriate language. Eventually, this language should be able to indicate the independent sections of a job that can be processed simultaneously. The PL/I language uses FORK and JOIN statements for indicating the possible parallel computation (8). However, this problem is not severe in the simulator described in this thesis, as the various physical files to be searched can be processed in parallel.

5.2 An Overview of the Simulator

The proposed architecture has been simulated at the PMS level. The simulator is written in FORTRAN IV. There are two characterizations required for this simulation:

(i) Characterization of the user group,

(ii) Characterization of the system.

The performance of the system is indicated by the output parameters like the response time and its statistics, the utilization of various resources and their statistics. Two more parameters of the simulator are identified as follows. The 'local parameters' are the attributes of the system being simulated. The number of processors, input channels and output channels are some examples. The "global parameters", also known as the "input parameters" characterize the operating environment. The arrival rate of users, average number of files to be searched per query and their distribution are some examples for the input parameters.

Input Parameters
Local Parameters
Output Parameters

Figure 5.1: Simulator

A complete list of the parameters used in this simulator is given in Table 5.1.

The overall organization of the simulator is shown as a block diagram in Figure 5.2. The names of various blocks correspond to the names of the subroutines simulating them (See Appendix IV). The arrival of a request to the system is simulated by the ARRVL routine which is then processed by the PROCES routine. The request is further characterized by the number of files to be searched to obtain the response for the query. This is done by the GENFIL routine. Allocation of jobs to the parallel processors is done by the PROCES routine with the help of the SCHED routine.. At the end of the processing, the OUTBUF routine is called which simulates the output processor as viewed by the parallel processors. The above sequence of operations simulate the processing of a single customer. A number of such arrivals are simulated

TABLE 5.1: List of Parameters in the Simulator

I. Input Parameters

(a) Characterizing the users:

1. Average¶ inter-arrival time (XMA)*
2. Average number of files per request (XMF)

(b) Characterizing the System:

1. Average processing time per core load (XMP)
2. Average storage access time (XMS)
3. Average output time per file (XMT)

(c) Simulation Parameters:

1. Identification
2. Number of simulation runs required
3. Interval limits for histograms
4. Output options
5. Seeds for random number generators.

II. Local Parameters

(a) Number of processors
(b) Number of input channels
(c) Number of output channels

III. Output Parameters

(a) Average waiting time (i.e. waiting time of a request to get atleast one freeprocessor for service).

(b) Average transit time (time between job entry and its completion is defined as the transit time)

(c) Average utilization of processors, input and output channels

(d) Load sharing by the concurrently operating elements like processors and channels.

¶ The averages have an associated probability distribution. For example, the inter arrival time is exponentially distributed with the given value of XMA as its mean.

* The names inside paranthesis indicate the FORTRAN variable names used in the simulator for these parameters.

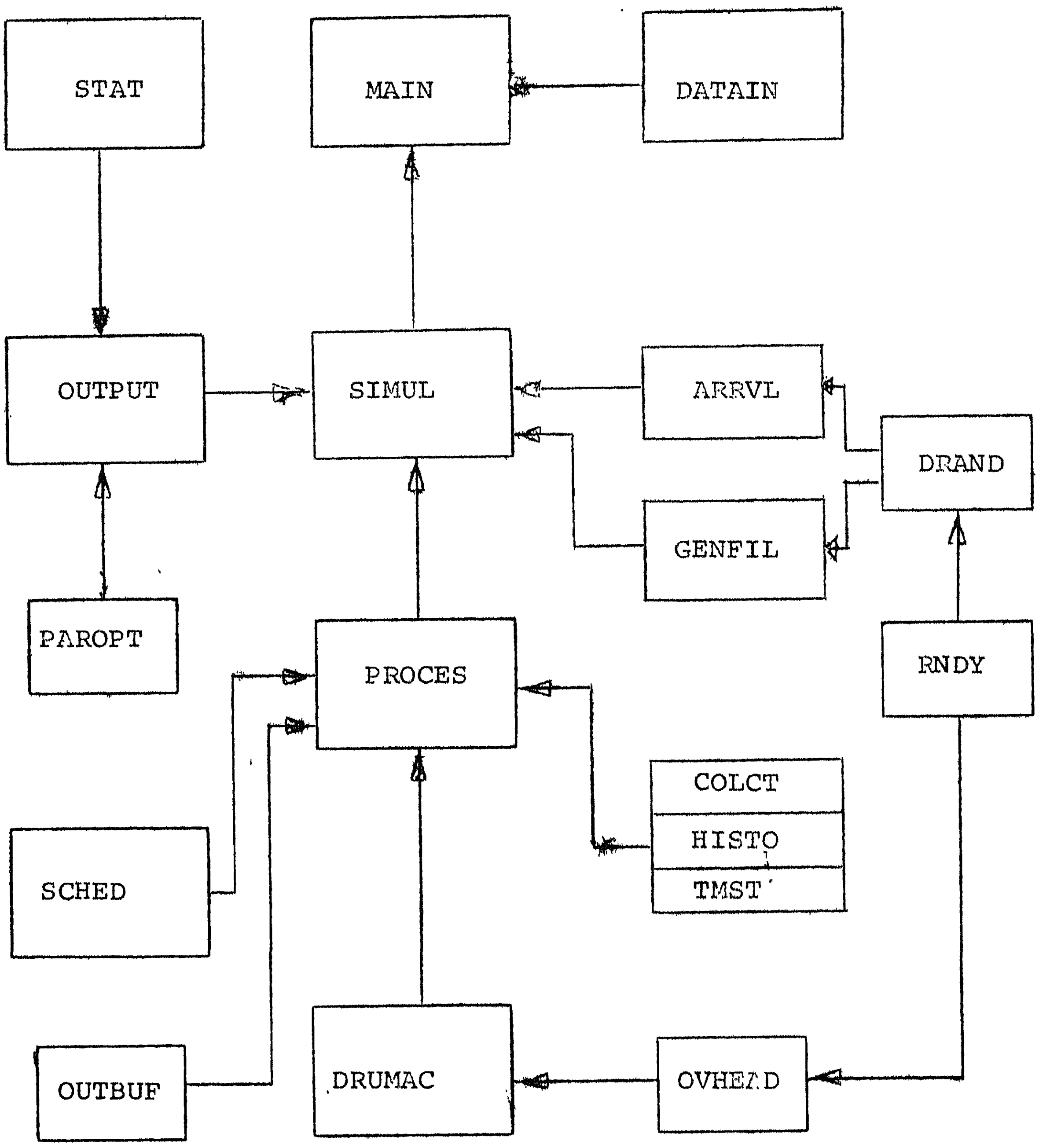


Figure 5.2: Organization of the Simulator

to constitute one simulation run and required statistics for output parameters are collected. The OUTPUT routine would print the various statistics of the output parameters in a suitable format (See Appendix V).

However, as the system is simulated to study the effect of varying local parameters, it is possible to have more than one simulation run for varying local parameters in the same computer run. To facilitate the plotting of the data obtained from various simulation runs, the PAROPT routine has been introduced.

As a single simulation run consists of the simulation of a few hundred customers (500 in the present case) and a number of such runs are required with different values for local parameters, it is necessary to code the simulator efficiently. For this purpose, an event based approach is used rather than the clocked sequence system (65). The simulation language like GPSS being slow compared to compiler type languages, FORTRAN has been chosen. Further, the ease with which modifications can be introduced in various parts of the simulator, debugging facility and the common use of the language have motivated the choice of FORTRAN IV. A problem oriented, FORTRAN based simulation language like GASP (83) tends to become slower due to its being general purpose and hence the above simulator has been directly written in FORTRAN.

One of the main problems with simulation is, collection of data. For instance, the simulation of the first level processor in this architecture would need data about the user response time at the terminals, the synonym and file directory table lengths, table look up techniques used with them and so on. Some of these data can be obtained only when the system is operational and the initial design needs a rough estimate of these parameters. In this simulator, the first level processors are not simulated at the function level and the literature is rich with such simulation (39). The requirements of a simulator for output processor are similar to that of the first level processor. Hence, their simulation is not included here. The second level processors, and the index file form the basis of simulation. The input to the simulator is the output of the first level processor as would be obtained in the total system. Similarly the output of the simulator corresponds to what would be the input to the third level processor in the total system.

The core of the architecture as simulated by this simulator is shown in Figure 5.3.

The system parameters to be determined with this simulator in the total system design are the number of processors, number of input channels and output channels (Figure 5.3). These three variables together are referred to as "local parameters" of the simulator. The response of the system for different values of the local parameters is studied with the help of the output parameters and their statistics. By observing the output of the simulator, the local parameters are suitably adjusted to obtain the desired transit time which might be a

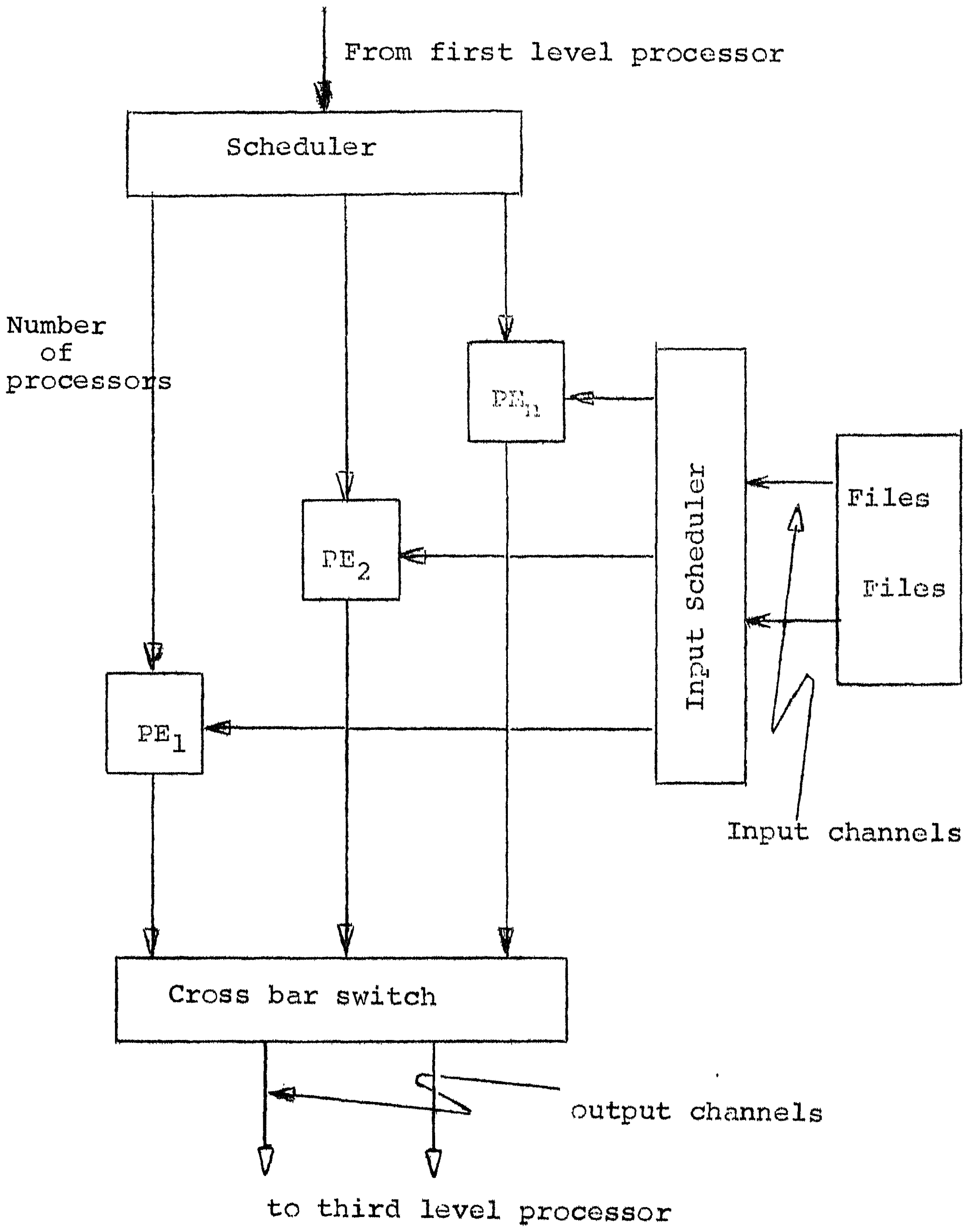


Figure 5.3: Part of the architecture as viewed by the simulator

design criterion.

An example for this iterative use of the simulator in system design is given in Section 5.4. Conversely, the simulator can be used to obtain the possible performance for the given resources, namely, the local parameter values.

It should be observed that the functional characteristics of the system are reflected in the system level simulation through the input parameters as the former is an abstraction of the latter. For example, the size of memory, processing algorithm and the instruction set of a second level processor are the factors which determine the expected processing time (XMP) in Table 5.1. It is possible to observe three categories among the input parameters.

(i) Input parameters characterizing the users or the operating environment.

(e.g.): Average arrival rate and its distribution

(ii) Input parameters characterizing the system at the PMS level as selected for simulation.

(e.g.): Average access time of the file; average processing time.

(iii) Control parameters; which control the options of the simulator itself. This is not part of the system but of the simulation program.

(e.g.): Print option; number of arrivals per simulation run.

In the following section a brief discussion of the various subroutines is presented.

5.3 Details of the Simulator

The simulator described in this Chapter is event-based. In discrete-event-based simulation, the operations of the system are considered at discrete points in time, called 'events'. Events cause the status of a system to change at a discrete point in time. An event is said to occur in this simulator when a query arrives. A system clock or master clock keeps track of the time of simulation. The time interval between two consecutive arrivals is obtained from a random number generator with exponential distribution. It has been found from experiments that the number of customers using an information system obeys a Poisson distribution (75). But the expected number of customers in a typical library vary with the time of a day and it is shown in Figure 5.4 for the Library at Indian Institute of Technology, Kanpur.

This distribution, further has a seasonal trend. In the simulation, however, neither the distribution over day nor the seasonal trend has been accounted for.

The built-in library functions, RNDY1, RNDY2, RNDY3, RNDY4 and RNDY5 are five independent pseudo random number generators with uniform distribution in the interval (0,1). The linear congruential method is commonly used for such random number generation (56). Suitable transformation (83) then gives exponential distribution, used for obtaining the inter arrival time, and the number of files required per query.

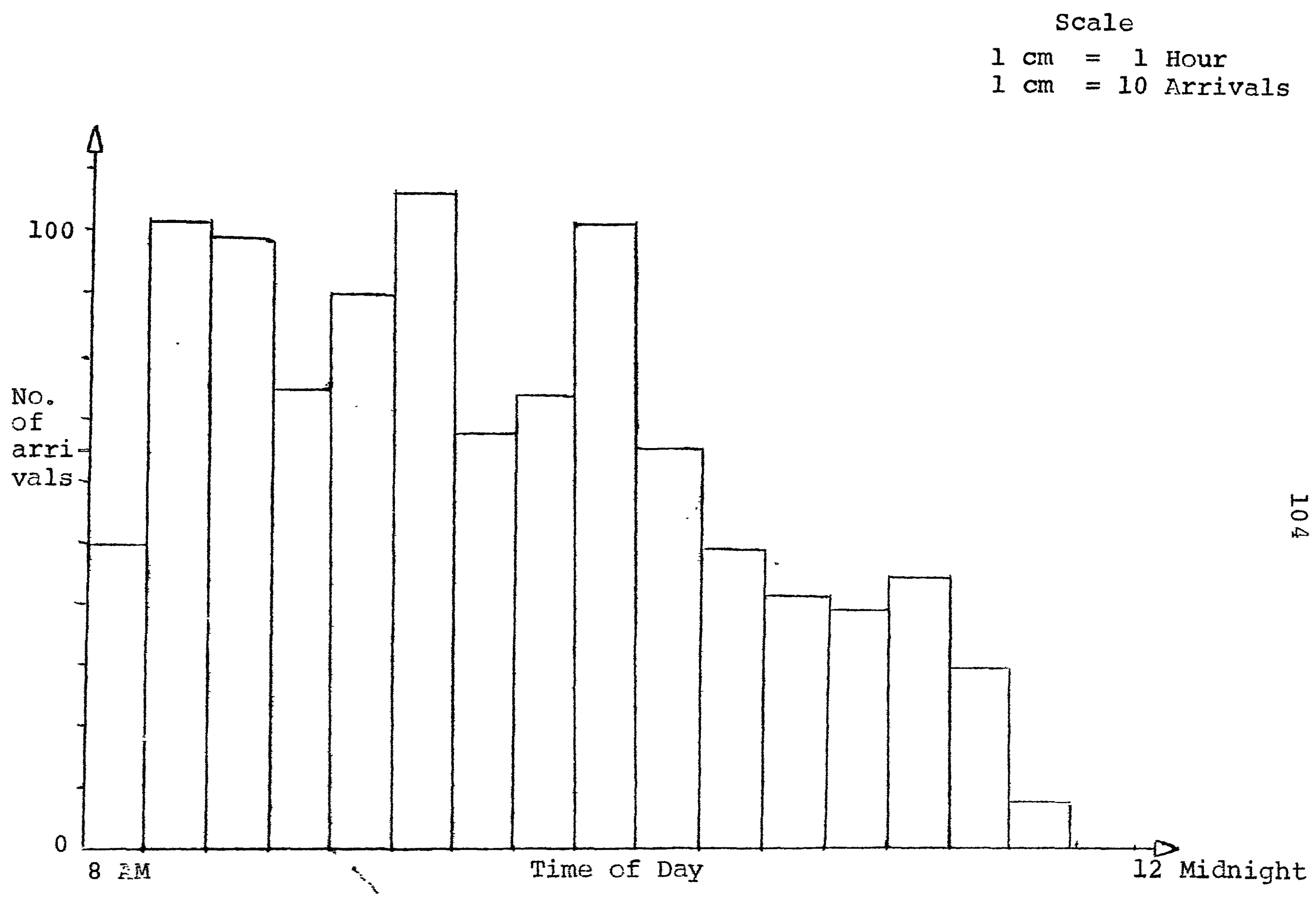


Figure 5.4: Typical user - arrival distribution.

Each parallel processing element in the simulator is associated with a local clock. For example, each processor has a processor clock and so also the channels. The local clock indicates the time after which that element will be free for new assignment. As pre-empting among users is not considered in this simulation the management of these clock times is much simpler. Consider the two conditions, namely, processor waiting for job and a job waiting for the processor as shown in Figure 5.5.

Suppose an event occurs at time t_N as in Figure 5.5(a) which means an arrival of a request. The processor is engaged until t_2 in processing the task assigned to it at time t_1. Starting from t_2 onwards the processor is waiting for a task which it may receive at time t_N. Then $t_N - t_2$ is the idling time of the processor. The Figure 5.5(b) exhibits a reverse condition, namely, the task waiting for a processor. The event occuring at t_N finds the only processor in the system, (assume there is only one processor) busy till t_m and has to wait for $(t_m - t_N)$ units of time to get a processor. Observe that system clock time minus the processor clock time is positive in one case and negative in the other case. This technique is used to simulate the concurrently operating components of the system. Each clock is a floating point number in the simulator program.

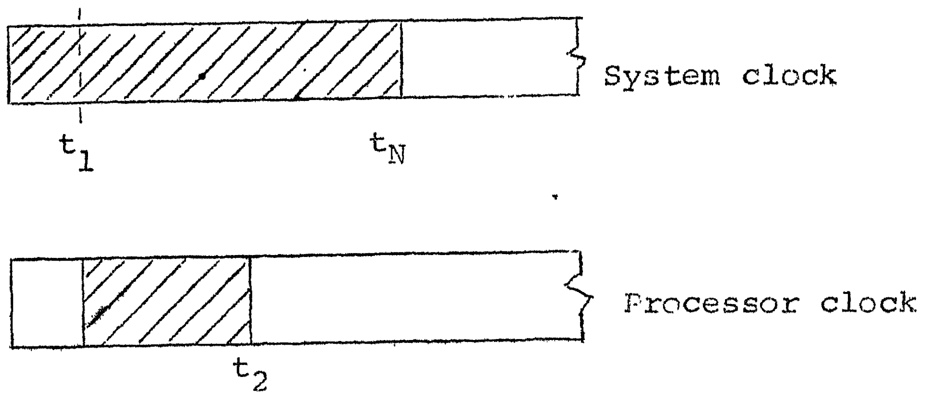


Figure 5.5(a) Processor waiting for job

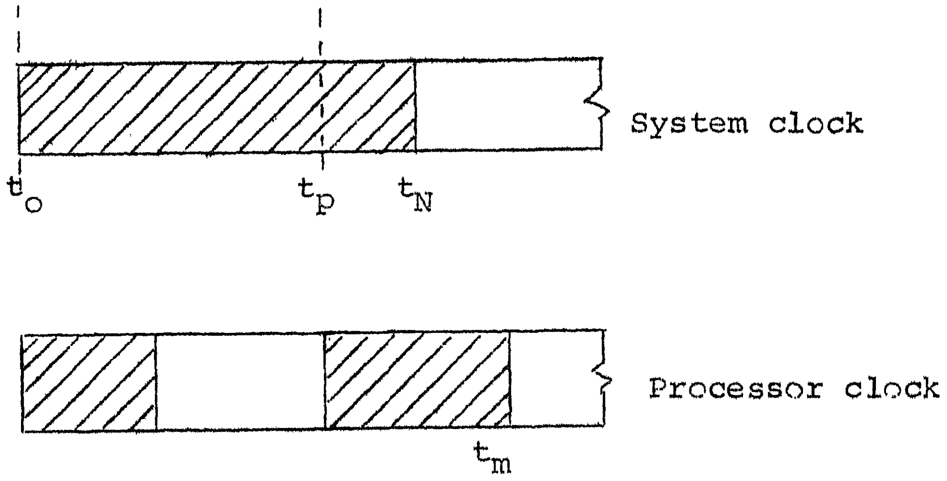


Figure 5.5(b): Processor busy condition

The asynchronous interrupt signals from the second level processors to the first level processor are simulated as explained below. A scheduling routine in the simulator looks at the clock times of the various processors and picks the first available processor to allot for a task at hand. In other words if t_{ci} is the clock time of the i-th processor and t_N the time of occurance of the event, k-th processor is alloted to the task such that

$$(t_N - t_{ck}) \leq t_N - t_{ci}$$
$$1 \leq i \leq M$$

when there are M processors in the system.
In this simulation the overhead involved in alloting the free processor to a task at hand is considered negligible.

As each event occurs, the status of the system is modified. The clocks associated with the processors and the selected channels are updated. The response time and waiting time at various stages are computed to collect required statistics on these parameters.

The index files might be stored on different types of physical files. The channels are capable of reading the physical files simultaneously. Any one channel can be connected to any one of the selected processors. The reading from a physical file has been assumed to be fixed to a single channel. The reading mode of the physical file may be conventional in which only one processor receives information from one channel. However by providing periodic synchronizing signals it is possible to use what are known as "rotary files" and a discussion of them is given in Section 5.6. When the physical file is larger than the memory of the processor the processor has to keep track of the sections of the files to be processed. All processors alloted to a query from the same terminal are connected to the same output line. The cross bar switch facilitates the connection of a processor to any of the output channels. This arrangement avoids any sorting that would otherwise be needed at the output level. Since more than one processor might be connected to the same output channel there is a possibility of queueing. The selected pointers are stored temporarily in its memory and the P.E. waits till the output channel is free for transmitting the pointers. The output interrupt suggested in Chapter IV for the second level processor would be useful in this context.

The operations discussed above are achieved in the simulator through a set of subroutines as shown in Figure 5.2. The following is a brief description of the essential sub- routines in the simulator.

MAIN:

This keeps track of the number of runs with different local or input parameters and calls the SIMUL routine for each set of parameter values.

DATAIN:

The input data provide values for various input and local parameters of the system and they are read in this subroutine. There are seven separate and logically independent types of data. Any possible subset of these can be read without affecting the values of the remaining parameters. All inputs to the simulator are in this routine.

SIMUL:

This simulates the request by updating the system clock (SCLCK) and generating the number of files to be searched for the query. Events are scheduled one after another in this routine until the termination conditions are satisfied either by the number of arrivals or by the time of simulation.

PROCES:

In this, the waiting time of the job or the processors are calculated and the processor clocks are updated. It in turn calls DRUMAC, for simulating the index file access and OUTBUF, for simulating the output conditions at the end of which, transit time of a job is calculated.

DRUMAC:

This subroutine simulates the index file access. Each channel has a clock associated with it. These channel clocks are used in the same way as the processor clocks for obtaining the channel idling time or waiting time of jobs for the input channels.

OUTBUF:

This is similar to the DRUMAC subroutine but used for output channels.

SCHED:

This subroutine allots the processors for tasks. In this simulator each physical file is allotted to one processor and the processor that becomes free is assigned to the task at hand. A job having more files to be searched, automatically gets more number of processors than others. It is possible to simulate other types of scheduling algorithms by modifying this subroutine (28).

In addition to the above major routines there are subroutines for generating random numbers of desired distribution, for collecting statistics on output parameters and printing the output in proper format.

The main objective of simulation is to study the effect of varying the local parameters on the output parameters to obtain an optimal design. Hence, there is a provision in the simulator for automatically varying the local parameters over a specified range. It would have been ideal to decide the next set of local parameters values after observing the output parameters for the current values at an interactive terminal. The limitations of the batch processing system* used for this simulation has motivated this scheme of automatic parameter changing in the simulator.

* IBM 7044/1401 computer system with IBSYS operating system.

5.4 Use of the Simulator in System Design

In this section, the output as obtained from the simulator and a method of design are discussed. Let the three local parameters, namely, number of processors, input channels and output channels be denoted as X1, X2 and X3 respectively. For each triplet of values X1, X2, X3 we get a set of values for output parameters. Consider the output parameter, transit time of jobs, for example. From the time a job enters the simulated system till it passes out, there are three queueing points. The job has to seize each of these facilities and be served till completion. Congestion at any one place will lead to an increase in transit time. Providing the facilities in large number is a solution but not optimal. One of the objectives of this simulation is to determine such an optimal selection for the given performance requirements and cost.

By allowing one of the three local parameters as a free parameter we obtain the output parameters as a function of this parameter. Figure 5.6(a) depicts one such function where the transit time is plotted as a function of the number of processors. The initial region of the graph indicates a processor (parameter X1) bounded condition. In this region increase in X1 results in decreased transit time. In region 2, the system is bounded by the fixed parameters, namely X2, X3 or both. The increase in X1 does not decrease the transit time. If the waiting time of processors for input and output channels are plotted as shown in Figure 5.6(b), the parameter which is responsible for saturation in region 2 can be identified. For instance, by observing Figure 5.6(b), the X2 parameter is found responsible for

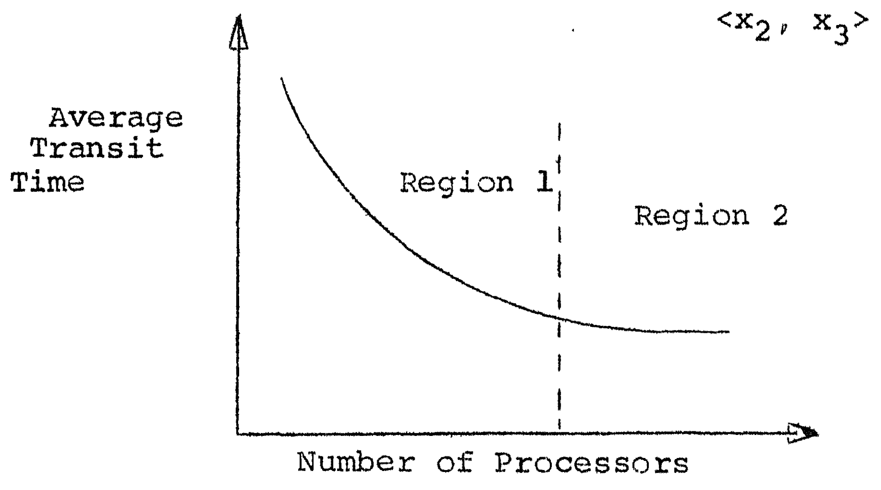


Figure 5.6(a) P.E. vs Transit time

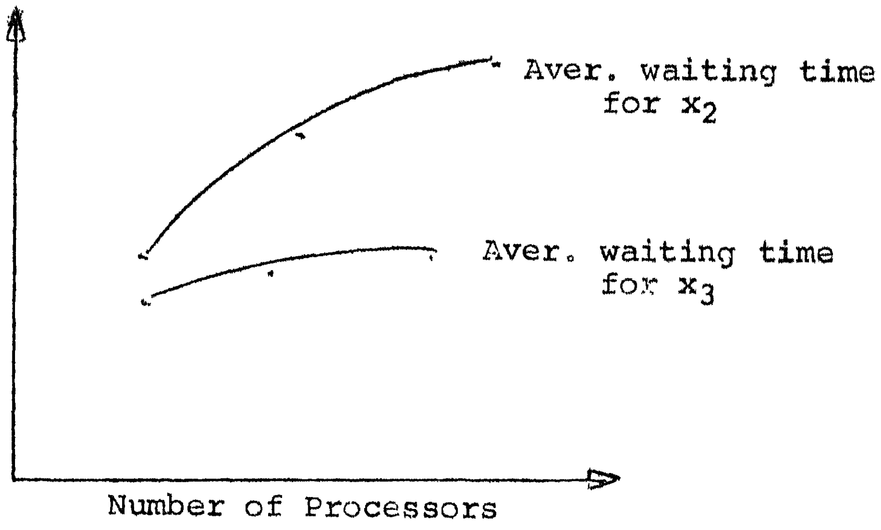


Figure 5.6(b): P.E. vs waiting time

saturation rather than X3 and we say it is input channel bounded. Any further decrease in transit time is possible by increasing X2, the number of input channels and not by mere increase in X1. The system design is an iterative experimentation with the simulator to obtain the least set of values for <X1,X2,X3> satisfying the requirements on transit time, cost and perhaps the utilization.

The local parameters, X1, X2 and X3 along three different axes and the output parameters, one along the fourth axis give a set of planes in four dimensions. The system design would be the selection of values for the local parameters from such plots.

However, such a plot even with five values for each local parameter would need more than two hours of computer time even with reasonably fast computer and simulator taking one to three minutes per simulation run. Thus it would be advantageous to record these performance curves for different input parameter values so that the recorded graphs will be useful as "design curves" for similar system design.

Observing Figure 5.6(a) we note that it may be approximated by an exponential function. An exponential approximation for such graphs obtained from the simulator are made using the least square criterion. Although the exponential approximation is good in region 1 [Figure 5.6(a)], it would be very much off the actual values obtained from the simulator in region 2. A nonlinear function approximation with exponential behaviour in region 1 and as a limiter in region 2 would be closer to the actual graph but difficult to use in analytical models. A linear polynomial fit with the logarithms of the ordinate gives the parameters of the exponential approximation, namely, amplitude and time constant.

The following formulae, obtained by the least square approximation (78) are used.

$$y = a_o + a_1 x$$

$$a_o = (K_o S_2 - K_1 S_1)/(S_2 S_0 - S_1^2)$$

$$A_1 = (K_1 S_0 - K_0 S_1)/(S_2 S_0 - S_1^2)$$

where

$$K_0 = \sum_{i=1}^{m} y_i$$

$$K_1 = \sum_{i=1}^{m} x_i \, y_i$$

$$S_o = m$$

$$S_1 = \sum_{i=1}^{m} x_i$$

and

$$S_2 = \sum_{i=1}^{m} x_i^2$$

Then

$$A = e^{a_o}$$

$$\tau = a_1 \quad \text{for} \quad y = A \, e^{\tau x}$$

The transit time t_r is a function of X1, X2 and X3 and can be written as

$$t_r = A \, e^{-\tau_1 x_1} \, e^{-\tau_2 x_2} \, e^{-\tau_3 x_3}$$

In general:

$$\tau_1 = f_1(x_2, x_3)$$
$$\tau_2 = f_2(x_1, x_3)$$
$$\tau_3 = f_3(x_1, x_2) \qquad (5.1)$$

where $f_1(.) f_2(.)$ and $f_3(.)$ stand for arbitrary functions. The requirement on the transit time can be expressed by the following inequality

$$A\ e^{-\tau_1 x_1}\ e^{-\tau_2 x_2}\ e^{-\tau_3 x_3} \leq 1 \qquad (5.2)$$

Each of the local parameters of the system is associated with a cost value which determines the total cost of the system. Then the following set of inequalities should be satisfied in system design:

$$C_{11}x_1 + C_{12}x_2 + C_{13}x_3 \leq C$$
$$A\ e^{-\tau_1 x_1}\ e^{-\tau_2 x_2}\ e^{-\tau_3 x_3} \leq P$$
$$x_1 \geq 1, \quad x_2 \geq 1, \quad x_3 \geq 1 \qquad (5.3)$$

The C_{1j}'s refer to the per unit cost of the processors and channels and C is the total allowable cost of the system.

As an example in iteratively using the simulator in system design, consider the following case. Table 5.2(a) gives the input data or values of the input parameters for this case study. Table 5.2(b) is the list of output parameters for some selected values of the local parameters, <X1, X2, X3>. The local parameter value <1,1,1> shows that the transit time is unacceptably high in an on-line system. Suppose, it is required to chose $<x_1, x_2, x_3>$ for an average transit time of

TABLE 5.2(a): Data for the case study

Input Parameters:

Mean time between two consecutive arrivals

= 60 secs * (XMA)

Mean number of files required per request

= 50 files (XMF)

(Distribution: Exponential)

Simulation Control Parameters:

Number of request per simulation run

= 500.

System Characteristics:

Number of files in the system

= 100

Mean time for processing one file

= 1.5 sec (XMP)

Mean time to access a file

30 sec (XMS)

Time spent in the output of pointers

= 2 to 5 percent of XMP.

* Names inside brackets correspond to the variable names in the program which identify them. See Appendix IV.

$<x_1,x_2,x_3>$	Mean Transit time	Mean waiting time for x_1	Utilization of x_1	Mean waiting time for x_2	Utilization of x_2	Mean waiting time for x_3	Utilization of x_3
<1 1 1>	863.3	798.3	87.4%	0	92.4%	0.0	2.90%
<3 1 1>	417.0	251.1	28.9%	103.1	91.6%	0.0	2.99%
<5 1 1>	416.6	167.6	17.4	186.5	91.6%	0.0	3.00%
<3 3 1>	87.6	48.4	37.8%	11.0	43.4%	8.9	3.98%
<5 3 1>	80.6	25.4	22.8%	38.6	43.4%	9.5	3.96%
<5 5 1>	35.6	9.7	23.8%	17.1	27.6%	7.1	3.92%

TABLE 5.2(b): Selected out$_{\wedge}^{put}$ parameters for a set of local parameters

about 35 seconds. From Figure 5.7(a) it is seen that increasing x_1 with fixed values for x_2, x_3 does not improve the transit time to any extent. The queueing or waiting time for x_2 is high and the number of channels needs to be increased. As observed from the Table 5.2(b) and Figure 5.7(a), it is clear that increasing any one parameter by itself is not sufficient.

Hence, the following procedure has been adopted in using this simulator for system design.

(i) Choose a trial value for $<x_1,x_2,x_3>$ and obtain a simulation run.

(ii) By observing the output parameter values, choose as the free parameter that local parameter value in front of which there is maximum congestion.

(iii) For different values of this free parameter observe the output of the system. Vary the free parameter until the congestion shifts to one of the remaining two local parameters.

(iv) At this stage, step 2 is repeated. When the output parameters reach the desirable design objective the procedure terminates.

By applying this iterative search procedure for the data given in Table 5.2(a) the minimum value for $<x_1,x_2,x_3>$ obtained is <5,5,1> and the output parameter values for this selection are shown in Table 5.2(b).

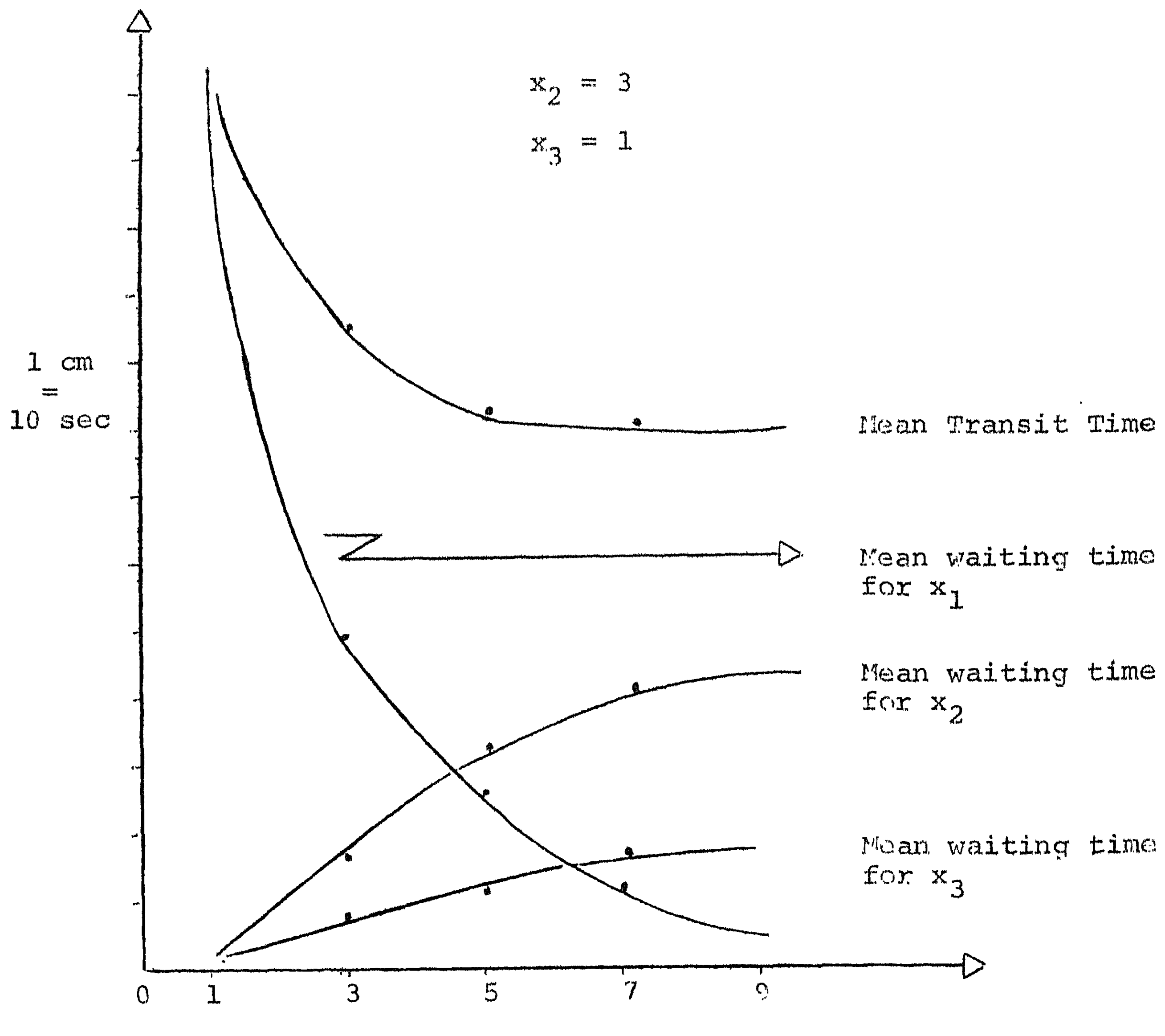


Figure 5.7(a): Transit time and waiting time vs P.E.

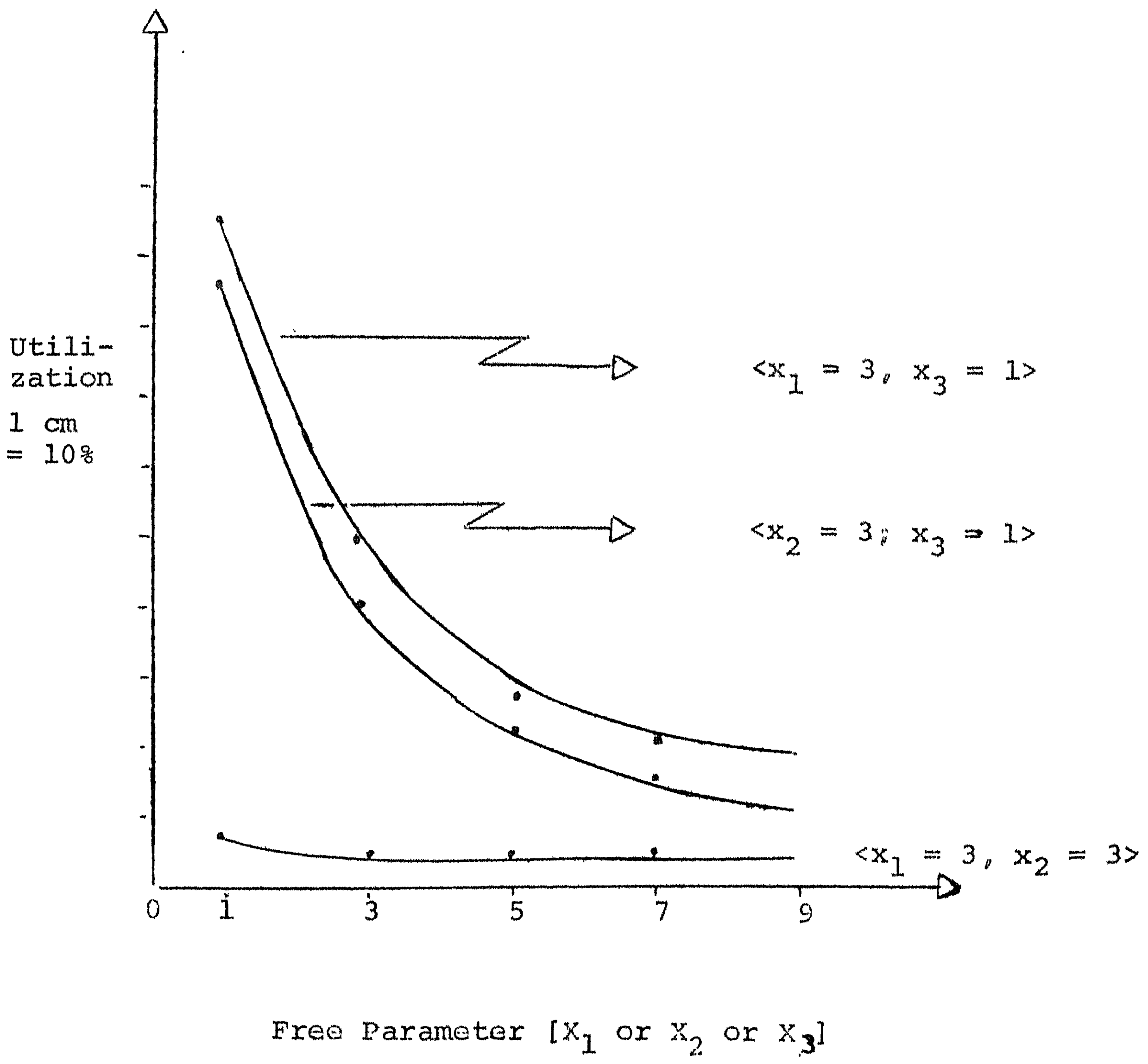


Figure 5.7(b): Utilization curve

The above iterative procedure is the one which searches for an optimum value for $<x_1, x_2, x_3>$ in a multi dimensional plane in order to satisfy the requirements on response time and perhaps on cost. The judicious choice of parameters for successive iterations would reduce the simulation time as compared to the brute force search over the entire plane. Observe that a brute force search by starting from $<1,1,1>$ is at least theoretically possible, as x_1, x_2 and x_3 are all integers and bounded. However, a good starting value for $<x_1, x_2, x_3>$ and on line decisions for successive trials would considerably reduce the simulation time.

The average utilization of processors and other parallel processing elements decreases as their number increases as shown in Figure 5.7(b).The average utilization of various resources when the constraints (Equation 5.3) are satisfied would be a measure of goodness for the system design. The decreasing utilization curves shown in Figure 5.7(b) also can be approximated with exponentials.

There are three different inequalities of interest to the system design both from the point of view of architectural and engineering considerations. They are:

(i) Transit time inequality called as response function,

(ii) Cost function,

(iii) Utilization function.

The utilization function could be a weighted sum of individual utilization of the resources and it would be an estimate of the running cost of the system. These three functions as obtained from the simulator can be approximated by analytical expressions. This suggests that an analytical method of approximately choosing the starting values for $<x_1, x_2, x_3>$ is possible. A discussion about this analytical model is presented in the next section. With this as the first estimate of the values for local parameters, the system design procedure would be as given in Figure 5.8.

5.5 Use of an Analytical Model

The requirements of an information system design is to select the set of optimal values for $<x_1, x_2, x_3>$, satisfying the cost and transit time constraints of Equation (5.4). With the cost and response functions as the two constraints and the utilization function as the objective function, it is possible to formulate the problem as a mathematical programming problem. If the three functions $f_1(.)$, $f_2(.)$ and $f_3(.)$ of Equation (5.1) are assumed to be constants, the response time inequality can be made linear. The following set of equations are obtained for the data given in Table 5.2.

$$C_1x_1 + C_2x_2 + C_3x_3 \leq C$$

$$2521.0 \; e^{-0.507x_1} \; e^{-0.392x_2} \; e^{0.25210^{-7}x_3} \leq R$$

$$\text{Maximize: } 40.9K_1 \; e^{-0.028x_1} + 38.9K_2 \; e^{0.015x_2} + 3.52 \; e^{-0.298*10^{-8}x_3}$$

$$x_1 \geq 1 \; ; \quad x_2 \geq 1 \; ; \quad x_3 \geq 1 \qquad (5.4)$$

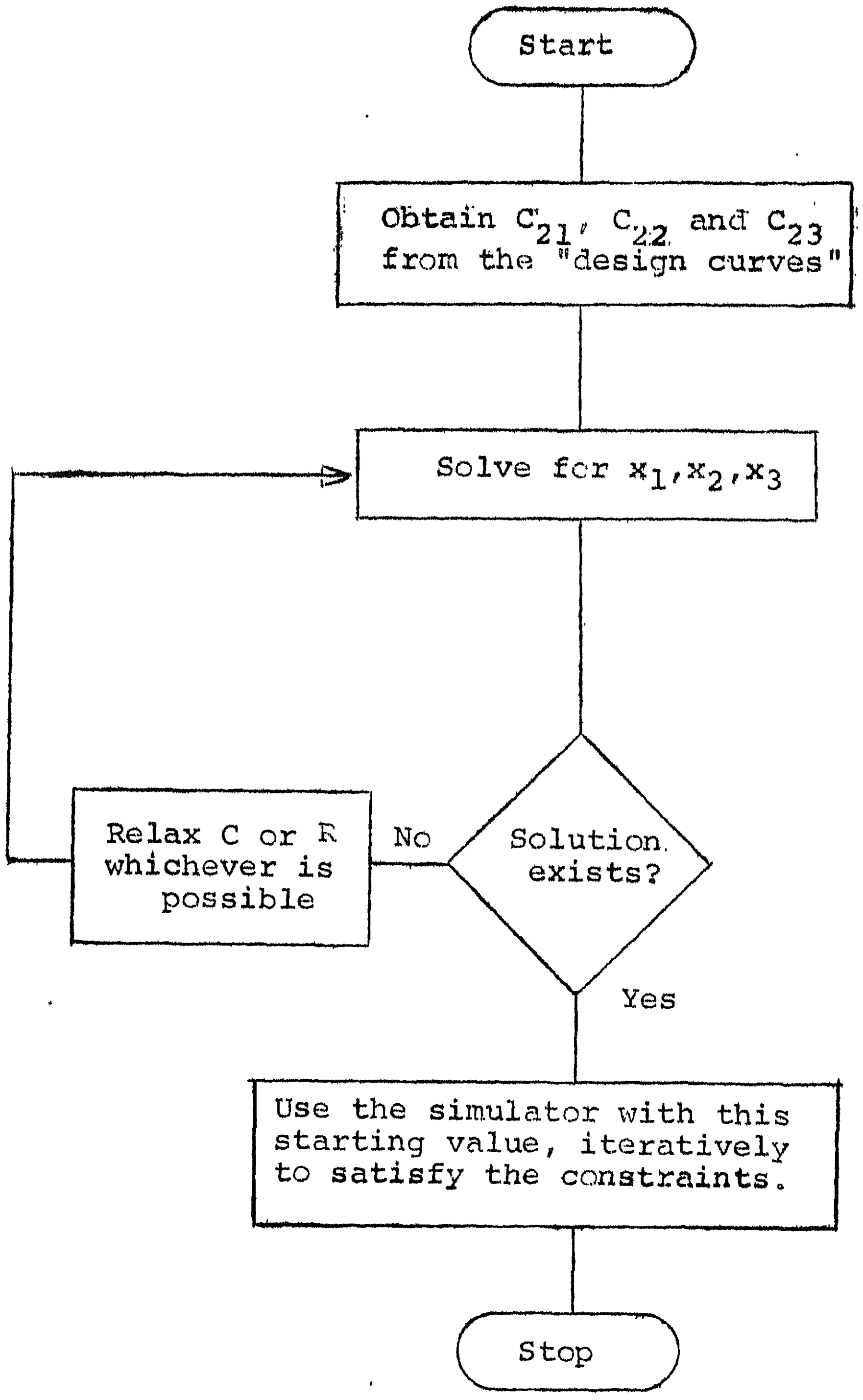

Start
Obtain C_{21}, C_{22} and C_{23} from the "design curves"
Solve for x_1, x_2, x_3
Solution exists?
No
Relax C or R whichever is possible
Yes
Use the simulator with this starting value, iteratively to satisfy the constraints.
Stop

Figure 5.8: Iterative design procedure

where $<C_1\ C_2\ C_3>$ are per unit cost vectors and the constants $<k_1, k_2, k_3>$ might depend on the cost vector. The different constants in these equations are obtained by approximating the graphs shown in Figure 5.7(a) and 5.7(b). This set of equations can be solved by the methods developed for nonlinear programming (41). By expanding the exponential functions in the objective function in Taylor series, it could even be solved as a linear programming problem (40). However, this approach has the following disadvantages.

(i) The distribution among $<x_1, x_2, x_3>$ is not accounted for. For example, the response time constraint in Equation (5.4) can be satisfied by fixing $<x_1, x_2>$ as $<1, 1>$ and increasing x_3 suitably. But, this is not true in practice. This difficulty arises mainly because the functions $f_1(.)$, $f_2(.)$ and $f_3(.)$ have been assumed to be constants.

(ii) The objective function is not very reliable and the sensitivity of the optimal solution for variations in objective function coefficients should be considered.

(iii) The variables $<x_1, x_2, x_3>$ themselves are basically integers. Solving the problem as a linear programming problem and rounding the solution vector to integer may not be valid.

Hence a straight brute force search technique has been used to find the starting value for the simulator from these equations. By considering the service times at the three queuing points in the simulation (See Table 5.2(a)) the following ratio between x_1, x_2 and x_3 has been chosen.

$$x_1 : x_2 : x_3 = 2 : 4 : 1 \qquad (5.5)$$

For different values of $<x_1, x_2, x_3>$ satisfying Equation (5.5), the response time from the analytical model is calculated until the required response time is obtained. If the combined cost of $<x_1, x_2, x_3>$ exceeds the total cost constraint before the response time constraint is satisfied there is no feasible solution. The values obtained for

cost vector	=	<6,5,1>
total cost	=	100
transit time	$\approx$	35 sec.

is $x_1, x_2, x_3 = <4,8,2>$

Values obtained from the simulation

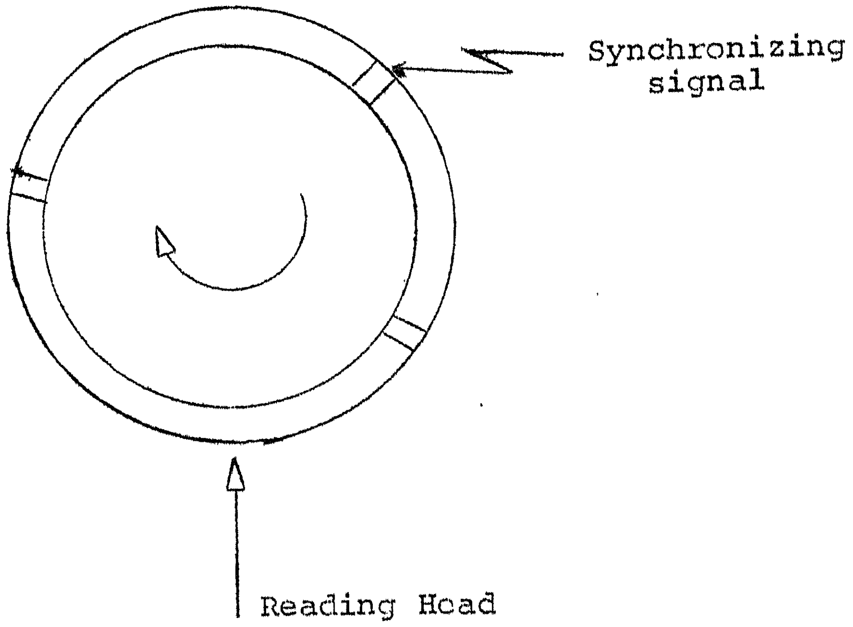
for the same transit time are: <5,5,1>

As the analytical model is an approximation to the simulation model these two values are not same. However using 4,8,2 as the starting value for the iterative use of simulator for design, would need much less number of iterations than otherwise.

5.6 Possible Extensions to the Simulator

A number of improvements are possible for this simulator. For example, consider the "rotary file" referred earlier in Chapter IV. The characteristics of this file is that the information contained in it is displayed constantly, whether it is required by the external system or not. Also assume that synchronizing signal, (possibly a fixed bit configuration) is given out by this file periodically. Any processor can start receiving the information with the start of the synchronizing signal and wait for one "cycle" to receive the complete information from this file. Using such files, it is possible to read the contents of a physical file for more than one processor simultaneously. As requests from the processors for file access are random, this provides a facility to start reading the file from any (logically correct) point and avoids the waiting time due to latency delay in the physical devices. Figure 5.9 explains the concept of a rotary file. The rotary file concept can be included in the simulator without any major change. Then the processors accessing the same file will not be serviced in sequence. Hence the waiting time of processors for channels will not depend on the service time required per customer but on the longest variable length record stored in the file. The two disadvantages of the rotary file systems are as follows: The synchronising code signal should not form a valid code either for the pointers or for the keys of various documents. Secondly, when the physical file size is larger than the memory of the P.E. the logic required to handle it in chunks is necessary for each processor, rather than for each channel.

When the number of files is more than the number of channels a facility to move the read/write heads is required. When such movable head devices are used it is possible to use this simulator in determining the scheduling algorithms and file organization which would minimize the lateral movements



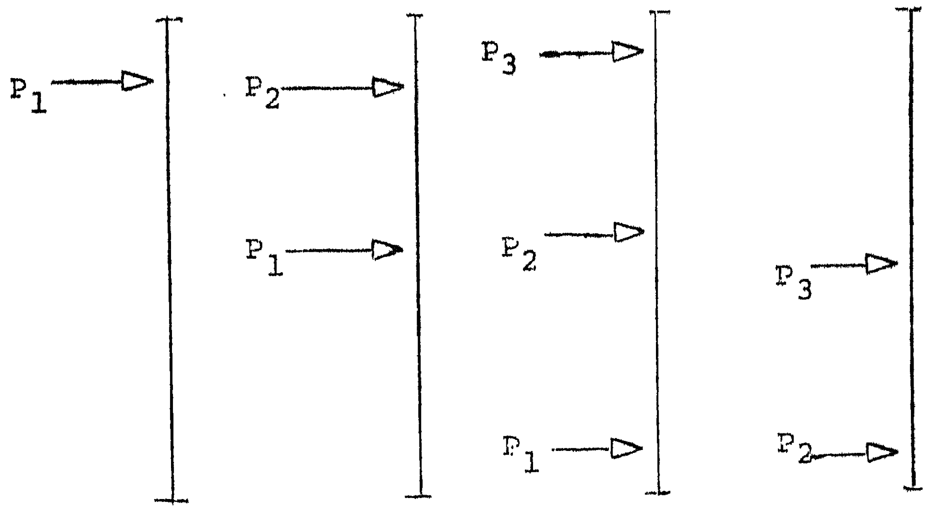


Figure 5.9: Rotary file.

(More than one processor can read segments of the file at the same time) of the head assembly.

The processor allocation algorithm used is first-come-first-served (FCFS). It is possible to experiment with other service disciplines like priority among services using this simulator.

The variation in the arrival rate as a function of the time of day can be simulated by considering the simulation over a period of time. An additional provision has been made in the simulator to control the simulation by the time of (not the computer time) simulation rather than the number of arrivals for this purpose.

CHAPTER VI

SUPERIMPOSED CODING AND THE ARCHITECTURE

6.1 Hardware Oriented Search

The computer architecutre discussed in Chapter IV consists of a hierarchy of processors in which the second level processors were designed for special application. Though it is a special purpose system proposed for information retrieval, it is not restricted to any indexing scheme or file organization. If a particular indexing scheme and file organization are assumed, then the search can be made faster and the cost of a typical second level processor would become that of a few gates and registers. Such second level processors have the program logic permanently wired-in and are less flexible.

By making the design highly special purpose, it is possible to make it faster at the cost of flexibility. For instance, the computer system discussed in Chapter IV can be used in any indexing system, whereas the hardware oriented design discussed below is not so. However the latter would be faster compared to the former.

The hardware oriented search is based on the coding scheme developed in (32) and the parallel access disk file (47). The key words of a document are coded and the information is packed into a word of sufficient length. The parallel access disk file has 64 tracks with a capacity of 4K words (16 bits) per track. Each track has a separate read/write head. The disk makes one revolution in about 30 milliseconds. During this period the information in the file is constantly being read into a register at a rate of about 2.18 mega bits/sec.

In the hardware oriented search discussed below, the coded binary vectors representing the documents would be stored on this parallel access disk file. This can be stored sequentially along the tracks or in "cylinder mode". In cylinder mode the bits of a binary vector are stored in the corresponding bits of different tracks and in this mode it is possible to read all the bits in this binary vector simultaneously. Hence it will be called "parallel mode" and the track mode of storing will be called "serial mode". Each of these methods have their own advantages and disadvantages. Before discussing the search methods, the coding scheme referred earlier is in sequel.

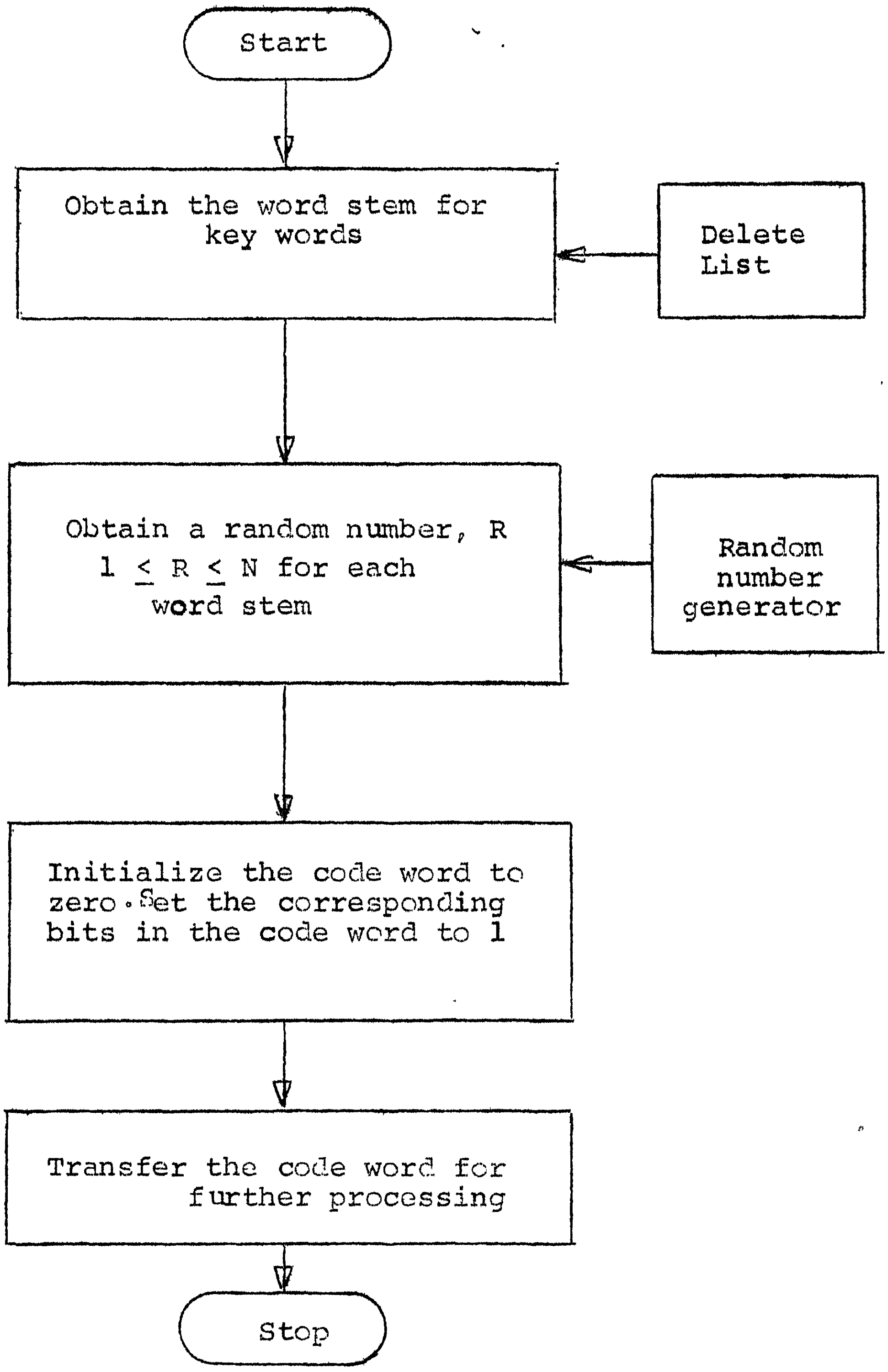
6.2 Superimposed Coding

In this method of retrieval each document is represented by a set of key words. Common endings such as s, ed, and compound endings like "fully" are removed. The resulting "word stems" are used in further coding. Each record is represented by a code word. Let there be N bits in this code word. From each word stem an integer is obtained by adding the numeric values of the letters in it. This integer is used to choose a random number from the uniform distribution of integers between 1 and N. If this value obtained from a random number generator is j, then the j-th bit in the code word is set to 1. Thus if there are n keywords ($n \leq N$), in an ideal case, there will be n ones in the code word and N-n zeros. However this may not be always true due to the following two reasons.

1. When the numeric values of letters are added, two different word stems having the same set of letters will give the same integer code.
2. The random number generator may give the same value in the range 1 to N for two different input integers. This again results in non-unique coding.

The spurious match due to non unique coding can be controlled. It has been shown that an optimum value for n is obtained when N = 2.2n where n is the number of keywords to be coded (32).

The given query is coded in the same way using the above algorithm as shown in Figure 6.1. Then each code word in the file is matched against the query code. A document is retrieved for the given query if the i-th bit of document code is one whenever the i-th bit of the query code is one for all values of i in the range 1 to N. Each document code has a pointer associated with it which refers to the excerpts in the text file. These pointers can be used in examining the text further.



Start
Obtain the word stem for key words
Delete List
Obtain a random number, R $1 \leq R \leq N$ for each word stem
Random number generator
Initialize the code word to zero. Set the corresponding bits in the code word to 1
Transfer the code word for further processing
Stop

Figure 6.1: Algorithm for superimposed coding.

When the probability of spurious matches is low the above method can be directly used for information retrieval. Otherwise this method is useful in rejecting the set of documents which are not relevant to the given query. The main motivation of this algorithm is to utilize the bitwise parallelism available in most of the modern computers for the inherent parallel processing possible in keyword oriented systems. In other words, query terms $q_1, q_2, \ldots, q_n$ are matched with the document terms $d_1, d_2, \ldots, d_N$ simultaneously.

However this superimposed coding scheme is inflexible. The indexing scheme is fixed and the query may have only Boolean AND operation on specified keywords. But using this scheme it will be possible to search a file of about 64K documents each with 30 descriptors in 30 milliseconds.

6.3 Bit serial and bit parallel methods of information storage

In both bit serial and bit parallel methods the following are the three stages in query processing:

(i) Query from the requester is converted to superimposed coding by the first level processor.

(ii) The indexed disk file and associated logic retrieves a set of pointers as response to this query.

(iii) The text pointers are used to access an "abstract file" for information display.

Bit Serial Method

In bit serial mode of storing the code words all bits of a code word are in the same track. Suppose a track has 4K bits storage and a code word and pointers are 32 bits long. Then it is possible to store 64 code words and their associated pointers in one track. The query code can be stored in a circulating register, Q. As each bit of the code word is read from the disk it is matched with the corresponding bit of the query code. At the end of the code matching the pointer is gated to the memory of the third level processor if the match is successful. Scuh matching logic can be provided for each track of the disk. Let q and d be n-bit query and document codes. Then the following Boolean expression M is 1 when they match

$$M = (\bar{q}_1 + d_1).(\bar{q}_2 + d_2).(\bar{q}_3 + d_3)\ldots(\bar{q}_{n-1}+d_{n-1}).(\bar{q}_n+d_n)$$

When the i-th bit of the query is 1 and the corresponding bit of the document code is 0 one of the Max terms is zero and so the function is zero. When the i-th bit of the query is 0 the corresponding maxterm is 1 irrespective of the document code bit being 1 or 0. This logic as used in matching is shown in Figure 6.2* for a typical track.

For the kind of disk discussed in Section 6.1, the interbit transfer time is of the order of 500 nano seconds. The matching logic response time could be made smaller than this to avoid any buffering. Typical integrated circuit gates have a delay time of 20-50 nano seconds (112).

* For the symbols used in this logic diagram see Appendix VI.

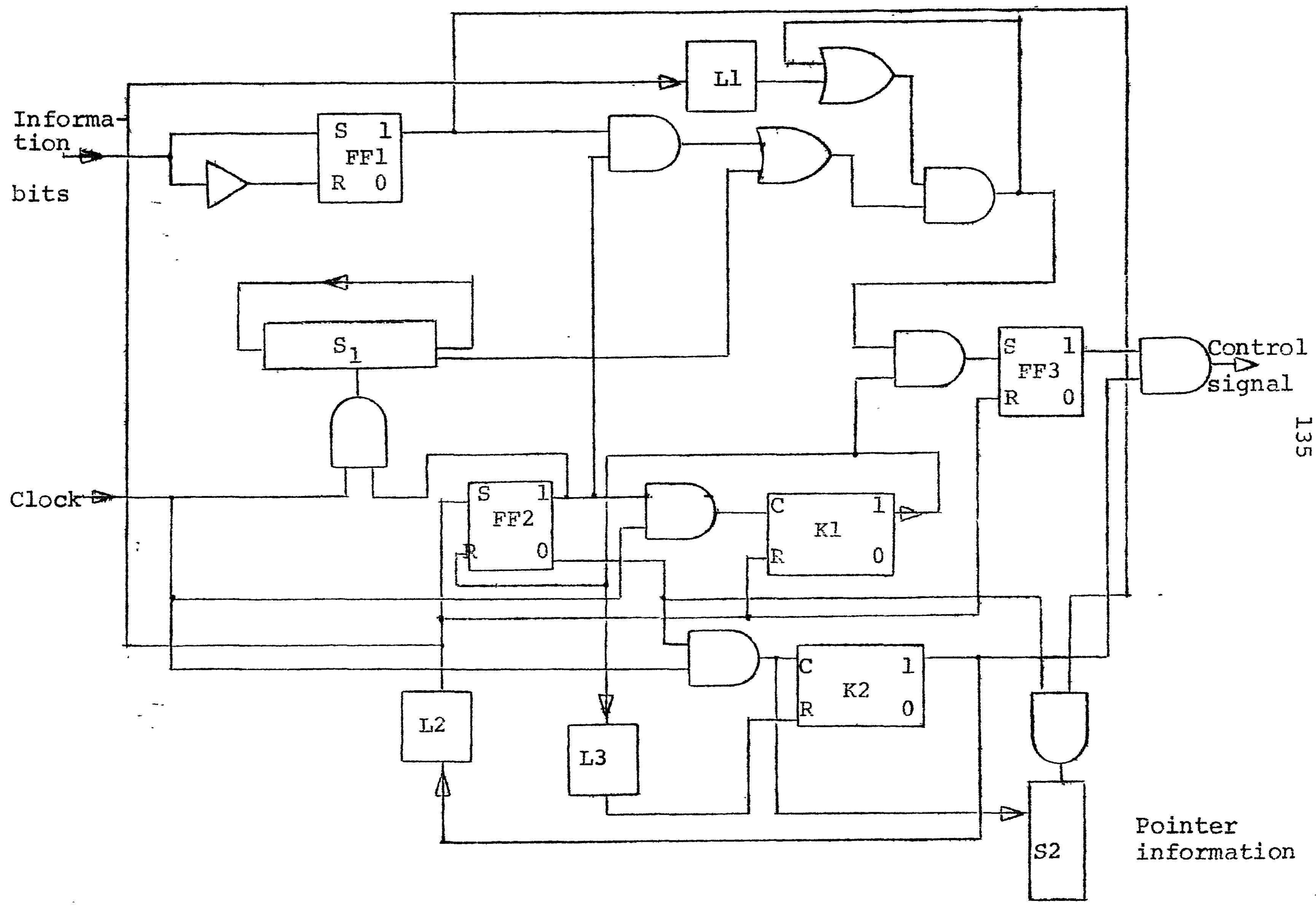


Figure 6.2: Matching logic

Legend of Figure 6.2:

Initial Status:

S1 register holds the complement query code $\bar{Q}$

FF2 initial set.

K1, K2 are initially reset.

Latch L1 is triggered.

$Z = 1.(\bar{q}_1+d_1).(\bar{q}_2+d_2)..(\bar{q}_n+d_n)$

FF1: is set to 1 or zero as 1 or 0 is read from disk (document code bit)

FF2: is a control flip flop

FF2 is 1 when document bits are counted

FF2 is 0 when pointer bits are counted

K1: Counter which counts n bits of the document code and also controls the FF2 to route the clock pulse either to K1 or K2.

K2: Counter similar to K1, counting the bits of the pointer information.

FF3: Holds the result of the Boolean expression Z above

L1: Latch supplying the Boolean constant 1 in the expression Z.

L2,L3: Reset and set latches.

The main difficulty in this bit serial mode is the need for concurrent memory access in storing the selected pointers. Suppose documents stored in two different tracks are matched and found relevant to the given query at the same time instant. Pointers from both these tracks will have to be stored in the memory before the selection of the next pointer. Under worst case the memory access time should be $n\tau/k$, where

n: number of bits in one code word plus that in a pointer

τ: interbit transfer time

k: number of parallel read and match logic

When $n = 64$; $\tau = 500$ n.s. and $k = 64$, a 300 n.s. memory would need no buffering. The cost of such fast semiconductor memories are comparable with convential core memories (4).

The typical diagram for bit serial method is shown in Figure 6.3. The query is held in the circulating register, S1. The counters K1 and K2 of Figure 6.2 can be common to more than one track depending on the fan out limitations. The black box named M refers to the matching logic of Figure 6.2 but excludes S1, K1 and K2.

The bit serial method has the following advantages:

1. The code word can be of any length and is restricted only by the capacity of the storage device.
2. Failure in one track will not cause the total failure.
3. Different tracks can store different lengths of code words and its use is discussed below.

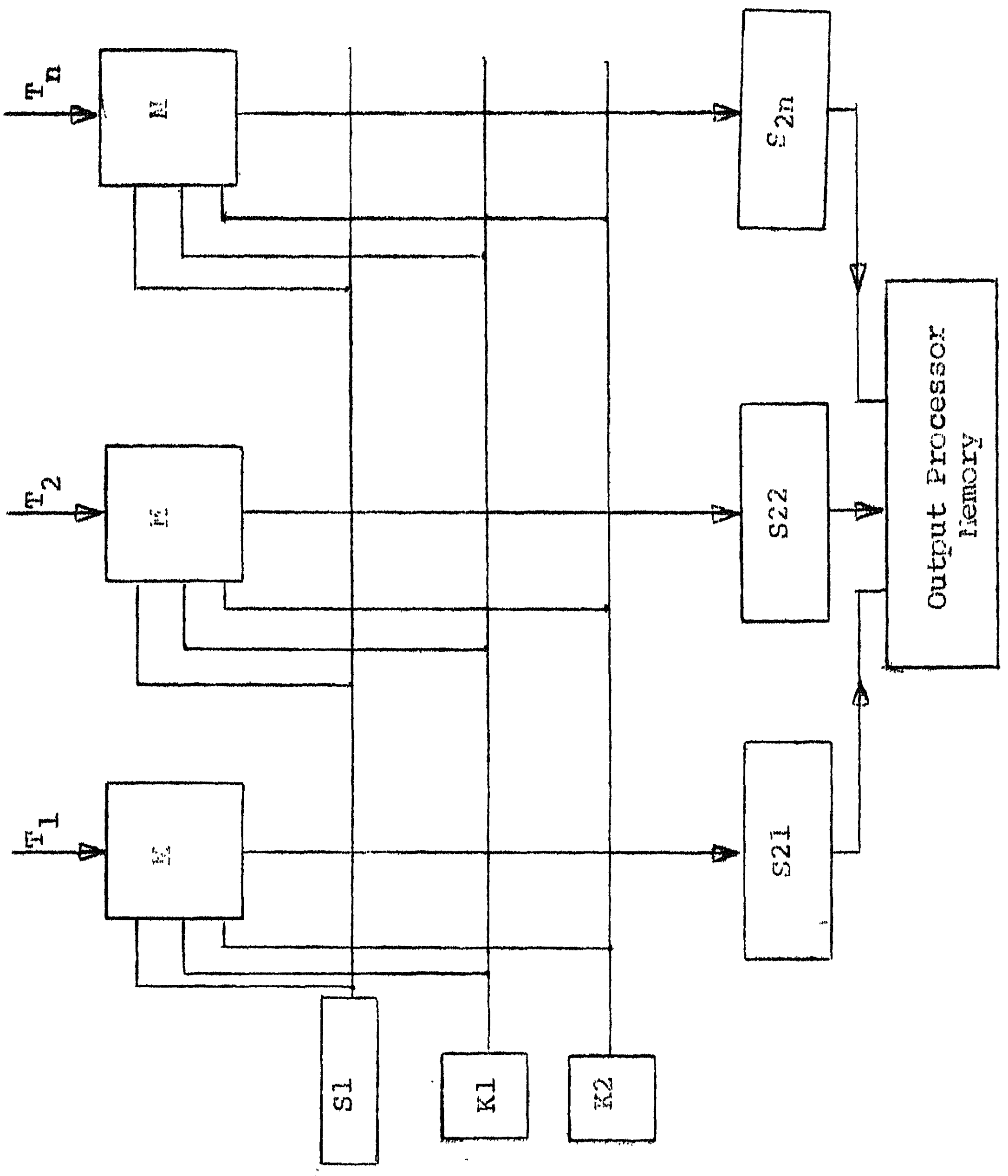


Figure 6.3: Bit serial matching

Consider the five attribute, ATNIC system discussed in Section 3.3. For survey and tutorial type documents the number of citations will be much more compared to their documents. This would need longer code words and using the same code word length for all documents may not be economical. In bit serial mode such code words can be stored on a separate track.

Bit Parallel Method

In this method all the bits of a code word are recorded in the cylinder mode and can be read at the same time. The query bits are matched with the document code bits all at the same time. If the match is successful the pointer associated with the document code word is selected. Suppose the disk has 64 tracks and the document code and the pointer together need less than 64 bits. Then it is possible to store one document along a radial line of the disc per bit position. When the track has capacity for 4K bits it is possible to store 4K documents.

Unlike the bit serial method, there is no buffering problem. Concurrent access to the memory of the third level processor will not occur in this method. However the disadvantage in this case would be lack of flexibility. To store documents with longer code words special logic is to be incorporated. One way to do this could be to use double length or triple length codewords and use one of the 64 bits to indicate such special codes. Such multiple length code words would need special attention in the preprocessing stage itself. Details of the logic required for bit parallel matching is shown in Figure 6.4. The m bit pointer information and the n bit document code are read in parallel into P and D registers (Figure 6.4). Then D register contents are matched with the query stored in the Q register in bit parallel. When the match is successful the P register contents are transferred to the memory. It is evident from the two figures that the logic of bit parallel method is much simpler. Comparison of the components required between bit serial method and bit parallel methods is presented in Table 6.1.

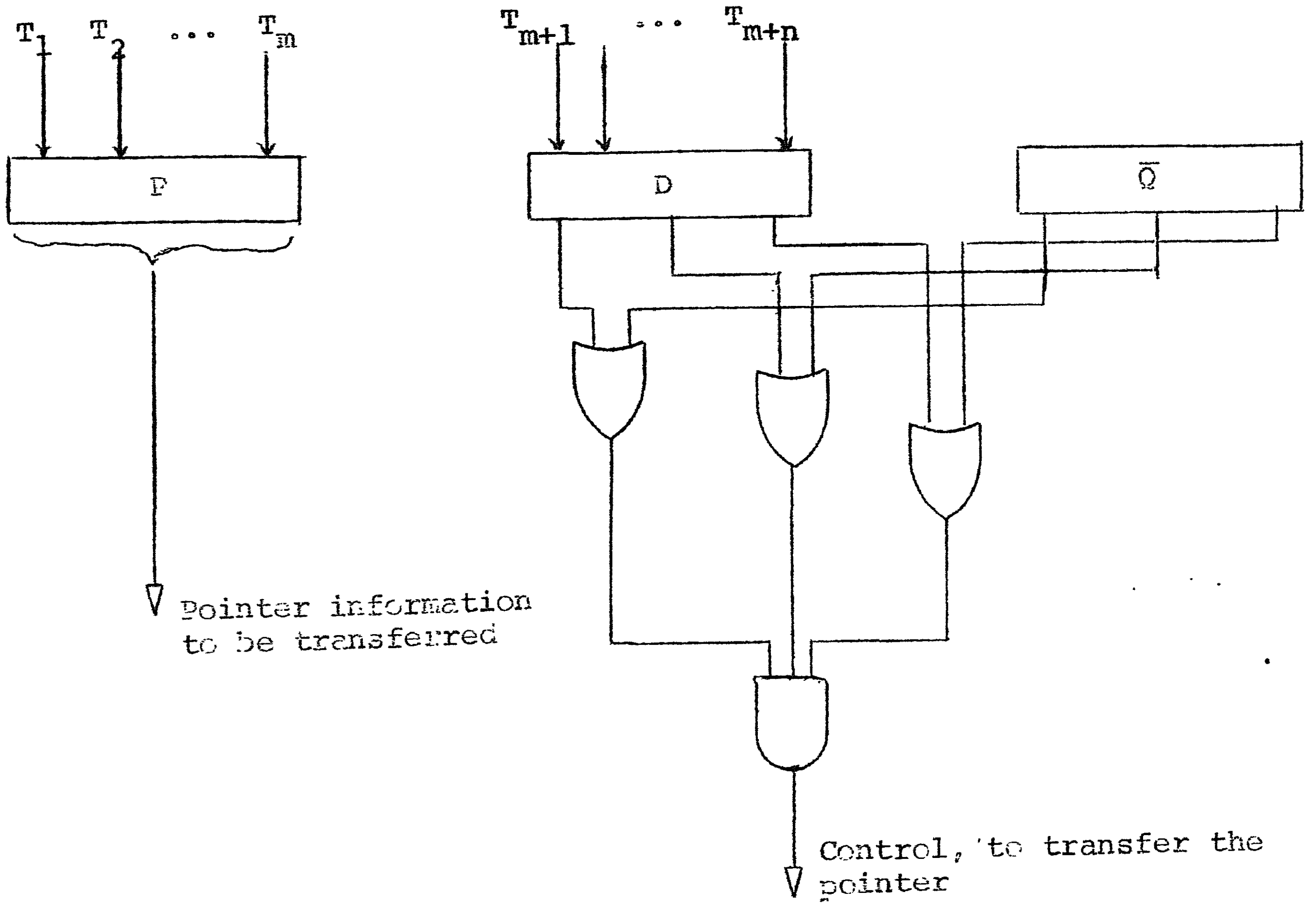
From the sample data used for ATNIC system referred to earlier, the following statistics have been obtained (the figures are rounded upwards).

Average number keywords/document : 10

Average number of authors/document = 2

Average number of citations/document = 7

However when indexing is more specific as much as 35 keywords per document is not uncommon (25). The main difficulty with this inflexible scheme is when there is need to handle such specific indexing. Using longer code word length of any length is not a solution. Note that the superimposed coding does not use storage efficiently. Further, representing the information in the form of connected graph or tree as used in SMART system is difficult to process with this scheme.



T_1
T_2
...
T_m
P
Pointer information to be transferred
T_{m+1}
...
T_{m+n}
D
$\overline{Q}$
Control, to transfer the pointer

Figure 6.4: Bit parallel matching

Component	Bit Serial	Bit parallel	
1. m bit registers	N	1	
2. n bit registers	2	2	N = No. of tracks
3. OR gates (Fan in 2)	2N	n	n = bits/code word
4. AND gates (Fan in 2)	5N+3	-	m = bits/ pointer
5. AND gates (Fan in 4)	-	n/4 + n/16 + ... 1	
6. NOT gates	1	-	
7. Counter modulo m	N	-	
8. Counter modulo n	1	-	
9. Flip flops	2N + 1	-	
10. Latch (mono stable multi)	N+2	-	
11. Fan out requirement of query register	N	1	
12. Fan cut requirement of co$\overset{un}{\cancel{n}}$ten$\cancel{t}$s	N	-	

TABLE 6.1: Comparison of the components required in bit serial and bit parallel methods.

6.4 Linear Query Processing

A query formulated with keys and Boolean operators is often known as Boolean query and the query with weights assigned to keys is called "linear query". The later has advantages in assigning relevance number to documents using the probabilistic indexing and the user assigned weights. From this point of view the above hardware oriented search is modified in this section to account for linear queries.

Suppose the weights assigned to various keys can be represented as a 4 bit binary number. Then the matching logic of each track in bit serial method will have the following additional functions to perform.

(i) When a non zero bit of the query matches with that of the document code, the corresponding weight is added to the accumulator.

(ii) When all the bits of the query code are matched with the document code, the one's complement of the threshold is added to the sum of the weights.

(iii) If there is overflow in the above addition, the pointer information is transferred to the memory for further display of this selected document.

The logic diagram shown in Figure 6.5 in conjunction with Figure 6.2 would achieve the above functions. W and τ registers hold the weight of keywords and threshold for selection respectively. W_i indicates the weight associated with the bit

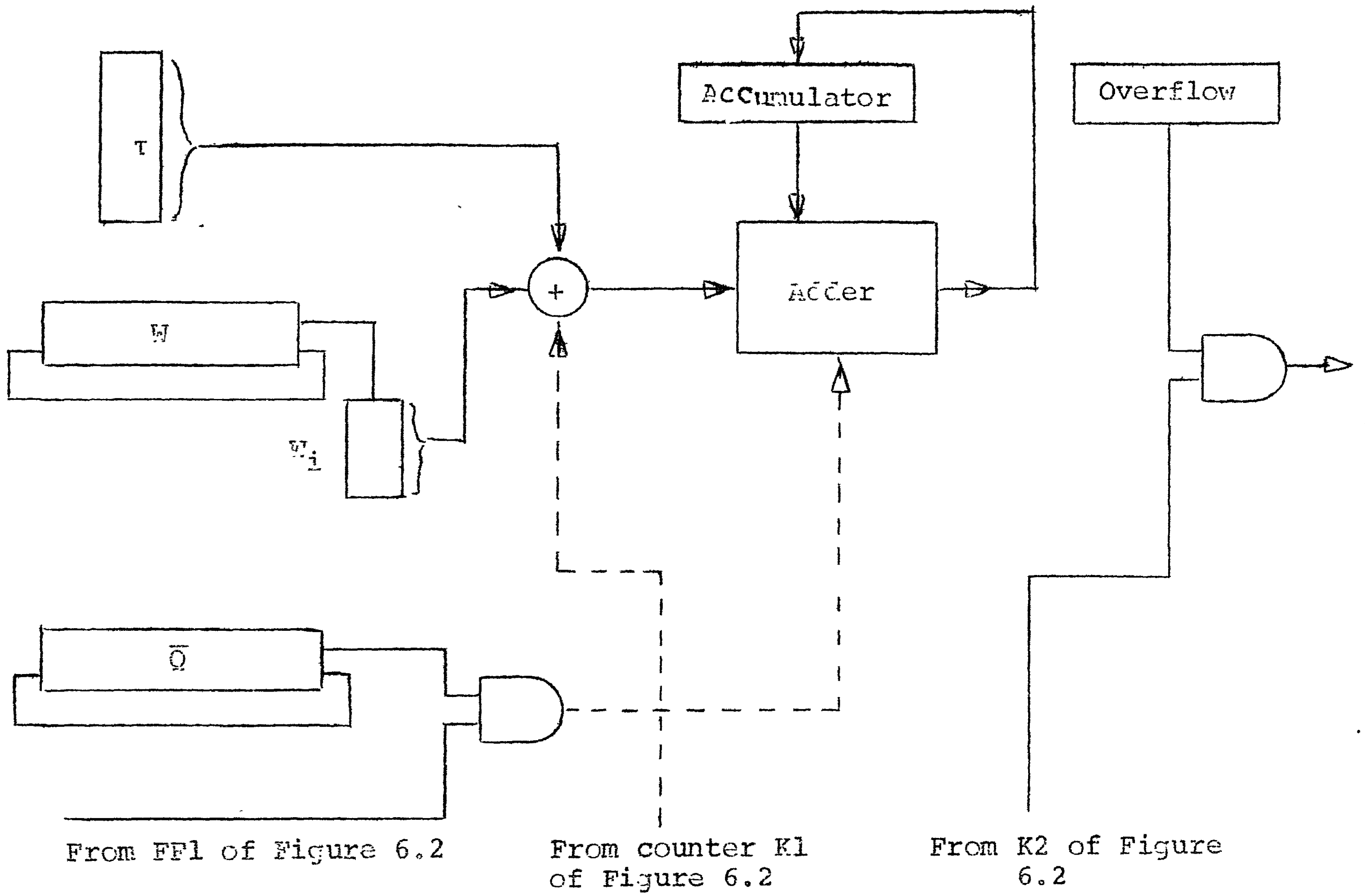


Figure 6.5: Linear Query Processing

being matched. At the beginning of each document code match, the accumulator is reset to zeros. Thereafter it keeps on accumulating the weights of the matching keys. At the end of document-query code matching, signalled by the counter K1 (Figure 6.2) the ones complement of τ is added to the accumulator and checked for overflow. If there is an overflow (which indicates the sum of the weights is greater than the threshold) the pointer information of this document that is buffered into S2 (Figure 6.2) is selected.

This modification would require an adder and an accumulator for each track and some additional registers. It is necessary to complete the addition process before the next bit is read. In linear queries if the weights of the keywords are constrained to be equal,it is possible to replace the adder by a simple one step counter.

In bit parallel match, as all the weights of the matched bits need to be added before the next bit arrives the situation is more tight. A pairwise addition of the selected weights would reduce the total time required for addition. Further, splitting the process of addition of weights into independent processes would be helpful when the addition process takes more time than the interbit transfer time. When processing time is large compared to the transfer rate and splitting the process into independent processes is not possible, the orgnization like the one described in Chapter IV will be useful.

The matching logic of Figure 6.2 is expanded in this section to the level of a processor capable of doing integer addition. The program for this processor is hard wired. It is possible to extend the power of this processor with an associated memory to the level of a "micro computer" as discussed in Chapter IV.

CHAPTER VII

CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

In this thesis, a multi-attribute system of information representation and its role in improving the performance of a retrieval system have been considered. An algorithm has been developed to pick a set of attributes to be used in organizing the given data base for retrieval. This algorithm when applied to a sample data in library information system gives keywords and the citations as the two important attributes (23). It has been shown that multi attributes and user feedback in retrieval system would statistically improve its performance.

A special purpose computer system, well matched to the problem of information retrieval has been proposed. This architecture takes advantage of the special nature of the information retrieval problems, namely, hierarchy of operations and inherent parallelism. A system level simulation of the parallel processing section of the architecture has been carried out in FORTRAN. The use of this simulator in selecting the number of parallel operating elements like processors and channels in order to meet the requirements of an information system in terms of the response time has been presented. An example for this design procedure with an iterative use of the simulator is worked out. The use of an analytical model in selecting the starting values for the free parameters of the simulation model has been discussed. The starting value obtained from the analytical model would reduce the number of iterations (in using the simulator) to arrive at the desired configuration.

By restricting to a particular type of indexing and coding it has been shown that it is possible to device a special purpose hardware oriented search. In this method, a set of few gates and registers together constitute a second level processor, when they are used with a parallel read disk. The fact, that highly specialized systems lead to a cheaper design at the cost of flexibility is reinforced by this. By incorporating elementary capabilities for addition and comparison into these processors, it is made possible to process linear queries with such specialized processors.

It is interesting as well as encouraging to note that the feasibility of designing such a special purpose system by observing that a mini computer of 8K memory (word length of 8 ~ 12 bits) for integer and Boolean data processing costs about five thousand dollars (11). A further decrease in hardware cost is possible with the recent developments in LSI (35). All these support the conclusion that the design of a parallel processing computer system with special applications to information retrieval problems would not only be technically feasible but also economically viable. This thesis presents a typical architecture for the same and the method of selecting the components of this architecture to meet the given requirements of an information system.

POSSIBLE EXTENSIONS

(i) The architecture proposed in this thesis will be useful in on line interactive retrieval. A query language development with due consideration to file organization and user interaction is worth investigating.

(ii) The reliability that can be obtained with the parallel processors in the second level and the modular nature of the system suggest its application in information networks.

(iii) A function level simulation of the P.E.s proposed in this system would be useful in improving the tentative selection of operation codes mentioned in this thesis. A detailed study of the various algorithms used in keyword matching systems and graph structured information matching will be useful in this context.

(iv) A complete simulation of the architecture with due considerations to the first and third level time sharing systems is interesting. The simulation model presented in this thesis will be part of such a total system simulation.

(v) Various constants used in the analytical model would depend upon the input parameters of the simulator like the arrival rate and average processing time per customer. Obtaining the required constants for the analytical model as functions of the input parameters and their study is interesting. But, this would involve considerable amount of computer time. However, such a study is worthwhile as this kind of computer architecture becomes popular.

(vi) Design and fabrication of the special purpose hardware oriented search discussed in this thesis is worth considering and will pose no new problems (113).

(vii) Design and testing of the complete architecture seems to be an interesting project particularly when it may be based on time sharing mini computers available in the market.

REFERENCES

1. Aron, J.D., (1969), "Information Systems in Perspective" Computing Surveys, V.1, p 213

2. Avedon, D.M., (1969), "An Overview of the Computer Output Microfilm Field", Proc. AFIPS, FJCC, V.35, p613.

3. Baker, F.B., (1962), "Information Retrieval Based on Latent Class Analysis", JACM, Vol. 9, p 512.

4. Barsamian, H., (1970), "Firmware Sort Processor with LSI Components", Proc. AFIPS, SJCC, Vol. 36, p 184.

5. Bar-Hillel, Y., (1963), "Is Information Retrieval Approaching a Crisis?", American Documentation, Vol. 14, p 95.

6. Bar-Hillel, Y., (1957), "A Logician's Reaction to Information Search System", American Documentation, Vol. 8, p 103.

7. Barnes, G.H., et al., (1968), "The Illiac IV Computer", IEEE Trans. on Computers, C-17, p 746.

8. Bates, F. and Douglas, M.L., (1967), "Programming Language ONE", (Book), Prentice-Hall, N.J.

9. Backer, J., and Hayes, R.M., (1962), "Information Storage and Retrieval ; Tools, Elements, Theoreis", (Book), John-Wiley.

10. Bell, C.G. (1969), "Computer Networks" in Computer Science Research Review., Carnegie Mellon University, Department of Computer Science.

11. Bell, C.G., et. al. (1970), "A New Architecture for Mini-Computers - The DEC-PDP 11", Proc. AFIPS, SJCC, Vol. 36, p. 657.

12. Bell, C.G., and Newell, A., (1970), "The PMS and ISP Descriptive System for Computer Structures", Proc. AFIPS, SJCC, Vol. 36, p. 351.

13. Bell, C.G., and Newell, A., (1971), "Computer Structures: Readings and Examples", (Book), McGraw-Hill Book Coy.

14. Bell, J.R., Bollinger, R.C., Jeeves, T.A., Mc rey nolds, R.C., and Shaffer, D.H., (1962), "On the Use of SOLOMON Parallel Processing Computer", Proc. AFIPS, FJCC, Vol. 22 p. 137.

15. Bird, R.M., and Fuller, R.H., (1965_, "An Associative Parallel Processor with Application to Picrute Processing", Proc. AFIPS, FJCC, Vol. 27, p 105.

16. Bobrow, D.G., (1964), "National Language Input for a Computer Problem Solving System", Ph.D. Thesis, MAC-TR-1, MIT.

17. Borko, H., and Beirmick, M., (1963), "Automatic Document Classification", JACM, Vol. 10, p 151.

18. Brown, J.S., and Reilly P.D., (1969), "The Use of Statistical Significance in Relevance Feedback" in Scientific Report No. ISR 16, p. IX-1., Cornell University, N.Y.

19. Buchholtz, W., (1962), "Planning a Computer System", (Book) McGraw-Hill Book Coy.

20. Buxton, J.N., (Ed)., (1968), "Simulation Programming Languages", (Book), North Holland Publishing Company.

21. Chang., W., (1970), "Single Server Queueing Process in Computing Systems", IBM Systems Journal, Vol. 9, No. 1, p. 36.

22. Chapin, N., (1969), "Common File Organization Techniques Compared", Proc., AFIPS, FJCC., Vol. 35, p. 413.

23. Chien, R.T., and Preparata, F.P., (1966), "Topological Structures in Information Retrieval" in Proc. Fourth Annual Alerton Conference on Circuit and System Theory, p. 359, University of Illinois, Urbana.

24. Cleverdon, C.W., (1962), "Report on the Testing and Analysis of an Investigation into the Comparative Efficiency of Indexing Systems", Crassfield England, The College of Aeronautics

25. Cleverdon, C.W., (1967), "The Cranfield Tests on Indexing Language Devices", ASLIB Proceedings, Vol. 19, No. 6, p173.

26. Constantine, L.L., (1969), "Integral Hardware/Software Design" (in nine parts), Modern Data Systems, Vol. 2, p 36.

27. Cros, R.C., Gondin, J.C., and Levy F., (1964), "SYNTOL - Syntagmatic Organization Language", Gauthier-Villars, Paris.

28. Denning, P.J., (1965), "Queueing Models for File Memory Operation", M.S. Thesis, MAC-TR-21.

29. Dunn, T.M., (1964), "Remote Computing - An Experimental System", Proc. AFIPS, SJCC, Vol. 25, p 413.

30. Edmundson, H.P., and Wyllys, R.E. (1961), "Automatic Abstracting and Indexing - Survey and Recommendations" CACM, Vol. 4, p. 226.

31. Estrin, G., and Fuller, R.N., (1963), "Algorithms for Content Addressable Memory Organization", Proc. Pecific Comp. Conference. March '63, p. 118.

32. Files, J.R., and Huskey, H. D., (1969), "An Information Retrieval System Based on Superimposed Coding", Proc. AFIPS, FJCC, Vol. 35, p 423.

33. Flores, I., (1966), "Computer Programming", (Book), p 333 Prentice-Hall, N.J.

34. Flynn, M.J., (1966), "Very High Speed Computing Systems", Proc. IEEE, Vol. 54, p. 1901.

35. Flynn, M.J., (1966), "A Prospectus on Integrated Electronics and Computer Architecture", Proc. AFIPS, FJCC, Vol. 29, p 97.

36. Fox, L., (1966), "Advances in Programming and non-numerical Computation", (Book), Pergamon Press.

37. Garfield, E., (1964), "Science Citation Index - A New Dimension in Indexing", Science, Vol. 144, p 649.

38. Gotlieb, G.C. and Kumar, S., (1968), "Semantic Clustering of Index Terms", JACM, Vol. 15, p. 493.

39. Greenbaum, H.J., (1969), "A Simulator of Multiple Interactive Users to Drive a Time Shared Computer System", M.S. Thesis, MAC-TR-58.

40. Hadley, G., (1962), "Linear Programming", (Book), Addison Wesley Publishing Coy.

41. Hadley, G., (1964), "Non-linear and Dynamic Programming", (Book), Addison Wisely Publishing Coy.

42. Hayes, R.M., (1963), "Mathematical Models for Information Retrieval" in Natural Language and the Computer. Garvin (Ed), McGraw-Hill Book Coy.

43. Hellerman, H., (1967), "Digital Computer System Principles", (Book), McGraw-Hill Book Coy.

44. Herscovitch, R., and Schreider, T.H., (1966), "GPSS III - An Expanded General Purpose Simulator", IBM Systems Journal, Vol. 4

45. Hosaka, M., and Tani, T., (1965), "A Real Time Computer System for Train Seat Reservation", Proc. IFIP Congress, Vol. 2, p 320.

46. Huskey, H.D., (1964), "An Introduction to Procedure Oriented Languages"in Advances in Computers, Vol. 5, p 349. Franz L. Alt and Morris Rubinoff (Ed), Academic Press.

47. Huskey, H.D., (1971), "Personal Communication About the Parallel Read Disk", Computer Centre, IIT-Kanpur.

48. Husson, S.S., (1970), "Microprogramming Principles and Practices", (Book), Prentice Hall.

49. Hutchinson, G.K., (1968), "Some Problems in the Simulation of Multiprocessor Computer System" in Simulation Programming Languages. J.N. Buxton (Ed) p 305, North Holland Publishing Coy.

50. Ivie, E.L., (1966), "Search Procedures Based on Measures of Relatedness Between Documents", Ph.D. Thesis, MAC-TR-29,MIT.

51. Kemney, J., (1962), "Libraries in 2000 A.D.", in Computers and the World of Future, Martin Greenberger (Ed)., MIT Press.

52. Kent, A., Belzer, J., Kurfeerst, M., Dym, E.D., Shirey, D.L., and Bose, A., (1967), "Relevance Predictability in Information Retrieval Systems", Method. Inform. Med. Vol. 6, p 45.

53. Kessler, M.M., (1963), "Bibliographic Coupling Between Scientif Papers", American Documentation, Vol. 14, p 10.

54. Kessler, M.M., (1965), "The M.I.T. Technical Information Project", Physics To Day, Vol. 18, No. 3, p. 28.

55. Khambata, A.J., (1969), "Introduction to Large Scale Integration", (Book), Wiley Interscience.

56. Knuth, D.E., (1969), "Random Numbers" in the Art of Computer Programming (Book), Addison Wesley.

57. [illegible]

57. Koczela, L.J., (1968), "The Distributed Processor Organization" in Advances in Computers, Vol. 9, p. 286. Franz, L. Alt and Morris Rubnoff (Ed), Academic Press.

58. Lancester, F.W., (1968), "Information Retrieval Systems", (Book), John Wiley.

59. Lancaster, F.W., (1968), "Evaluation of the MEDLARS Demand Search Service", U.S. Department of Health, Education and Welfare, Washington, D.C.

60. Lee, C.Y., (1963), "A Content Addressable Memory Applications to Information Retrieval", Proc. IEEE, Vol. 51, p 924.

61. Lehman, M., (1966), "Serial Mode Operation and High Speed Parellel Processing" in Information Processing, p. 631, Proc. IFIP, N.Y.

62. Licklinder, J.C.R., (1965), "Libraries of the Future", (Book), M.I.T. Press

63. Locke, W.N. (1970), "Computer Costs for Large Libraries", Datamation. Vol. 16, February, p 69.

64. Luhn, H.P., (1959), "Selected Dissemination of Information with the Aid of Electronic Processing Equipment", IBM Advanced System Development Division.

65. Markowitz, H., Hansner, B., and Karr, H., (1963), "SIMSCRIPT - A Simulation Programming Language", (Book), Prentice-Hall.

66. Maron, M.E.,and Kuhns, J.L., (1960), "On Relevence, Probablistic Indexing and Information Retrieval", JACM, Vol. 7, p. 216.

67. Maron, M.E., (1961), "Automatic Indexing on Experimental Inquiry", JACM, Vol. 8, p. 404.

68. McCoy, E.K., (1967), "The Ohio Bell Business Information System", Proc. AFIPS, SJCC, Vol. 30, p. 433.

69. McNeil, J.W., and Wetherell, C.S., (1969), "Bibliographic Data as an Aid to Document Retrieval", in Scientific Report No. ISR-16, Cornell University, Department of Computer Science.

70. Meetham, N.R., (1969), "Communication Theory and Evaluation of Information Retrieval System", Infor. Stor. & Ret., Vol. 5, p. 129.

71. Mendelson, M.J., and Englard, A.W., (1966), "The SDS Sigma 7: A Real Time, Time-sharing Computer", Proc. AFIPS. FJCC, Vol. 29, P. 51.

72. Morrier, R.E., Osborne, T.E. and Cochran, D.S., (1971), "The HP Model 9100A Computing", in Computer Structures, Reading and Examples. C.G. Bell and A. Newell (Ed), p 243. McGraw Hill Book Coy.

73. Montijo, R.E., (1967), "California D.M.V. Goes on Line", Datamation, Vol. 13, p. 31.

74. Mooers, C.N., (1959), "The Next Twenty Years in Information Retrieval : Some Goals and Predictions", Proc. of WJCC, p81.

75. Morse, P.M., (1968), "Library Effectiveness", (Book), MIT Press

76. Mueller, M.W., (1959), "Time, Cost and Value Factors in Information Retrieval", Presented at the IBM Information Retrieval System Conference. Pough Keepsie, N.Y.

77. Newell, A., Tonge, F.M., Veigenbaum, E.A., Green, B.F., and Mealy, G.H., (1961), "Information Processing Language - V. Manual (Book), Prentice-Hall.

78. Neilson, K.L., (1956), "Methods in Numerical Analysis" (Book) p. 264. The MacMillon Coy., N.Y.

79. Overhage, F.J., and Harman, R.C., (1965), "INTREX Planning Conference 1965", (Book), M.I.T. Press.

80. Pardee, D.R., (1961), "American Airlines SABRE Electronic Reservation System", Proc. Western Joint Computer Conference, May 9-11, p. 593.

81. Petschauer, R.J., (1970), "Trends in Memory Element and Subsystem Design", Computer IEEE Publication, Vol. 3, p 12.

82. Pcintel, N., and Cohen, D., (1967), "Computer Time-Sharing - A Review", Electrical Communication, Vol. 42, p. 193.

83. Pritskar, A.B., and Kiviat, P.J., (1969), "Simulation with GASP II", (Book), Prentice-Hall, N.J.

84. Prywes, N.S., (1963), "The multi list Central Processor" in 1962 Workshop on Computer Organization, (Book), p 214 Alan A. Barnum and Morris. A. Knapp (Eds), Sparton Book Company.

85. Prywes, N.S., and Gray, H.J., (1963), "The Multi List System for Real Time Storage and Retrieval" in Information Processing, 1962, p 273, Cicely M. Popplewell (Ed), Proc. IFIP.

86. Radhakrishnan, T., (1968), "Associative Memories", Lecture Notes for EE 663, Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India.

87. Radhakrishnan, T., and Rajaraman, V., (1970), "Compressed Index Terms and Threshold Matching in Information Retrieval" The Journal of Comp. Soc. of India, Vol. 1, p 35.

88. Rajaraman, V., and Radhakrishnan, T., (1971), "Principles of Digital Computer Design", Lecture Notes for EE 562, Dept. of Electrical Engineering, Indian Institute of Technology, Kanpur, India.

89. Ranganathan, S.R., (1965), "A Descriptive Account of the COLON Classification", (Book), Asia Publishing House.

90. Ranganathan, S.R., (1964), "Colon Classification" in Rutger's Seminars on Systems for the Intellectual Organization of Information, New Brunswick, N.J.

91. Raphael, B., (1964), "SIR : A Computer Program for Semmatic Information Retrieval", Ph.D. Thesis, MAC-TR-2, M.I.T.

92. Richards, R.K., (1966), "Electronic Digital Systems", (Book) p 262, John Wiley.

93. Salton, G., (1963), "Associative Retrieval Using Bibliographic Information", JACM, Vol. 10, p 440.

94. Salton, G., (1964), "A Document Retrieval System for Man-Machine Interaction", Proc. 19th Nat. Conf. ACM, p L2.3-1.

95. Salton, G., (1966), "Information Dissemination and Automatic Information System", Proc. IEEE, Vol. 54, p 1663.

96. Salton, G., (1968), "Automatic Information Organization and Retrieval", (Book), McGraw-Hill Book Coy.

97. Sammet, J.E., (1969), "Programming Languages : History and Fundamentals", (Book), p 603, Prentice-Hall, N.J.

98. Sander, W.B., (1968), "Semiconductor Memory Circuits and Technology, Proc. AFIPS, FJCC, Vol. 33, p 1211.

99. Schwartz, J.I., (1964), "Introduction to the System Development Corporation Time Sharing System", SDC Document, SP-1722, September 14.

100. Schwartz, J.I., Coffman, E.C., and Weissman, C., (1964) "A General Purpose Time Sharing System", Proc. AFIPS, SJCC, Vol. 25, p 397.

101. Senko, M.E., (1969), "File Organization and Management Information System" in Annual Review of Information Science and Technology, Vol. 4, p 113, Carl. A. Cuadra (Ed). Encyclopaedia Brittanica

102. Senzig, D.N., and Smith, R.V., (1965), "Computer Organization for Array Processing", Proc. AFIPS, FJCC, Vol. 28, p 117.

103. Shannon, C.E., and Weaver, W., (1949), "The Mathematical Theory of Communication", (Book), University of Illinois Press, Urbana.

104. Sharpe, W.F., (1969), "The Economics of Computers", (Book) p 495, Columbia University Press.

105. Simmons, R.F., (1965), "Answering English Questions by Computer : A Survey", CACM, Vol. 8, p. 53.

106. Slotnick, D.L., Borch, W.C., and McReynolds, R.C., (1963), "The SOLOMON Computer - A Preliminary Report", in Workshop on Computer Organization. C.A. Knapp (Ed)., Sparton Book Company.

107. Syn, W.M., and Lineberger, R.N., (1966), "DSL/90 A Digital Simulation Program for Continuous System Modeling", Proc. AFIPS, SJCC, Vol. 28, p 165.

108. Varian Data, (1968), "Varian Data 620/I Computer Manual", 216 Pico Boulevard Santa Monica, Calif. 90405.

109. Vickery, B.C., (1968), "The Raw Material of Retrieval" in Mechanized Information Storage, Retrieval, and Dissemination, (Book), p 15. Kjell Samuelson (Ed). North Holland Publishing Coy.

110. Waltson, C.E., (1966), "Information Retrieval" in Advances in Computers, Vol. 6, Franz L. Alt and Morris Rubinoff (Eds), Academic Press, p. 1.

111. Weinberg, A., (1967), "Science, Government and Information" in The Growth of Knowledge, (Book), p45. Manfred Kochen (Ed), John Wiley.

112. Wickes, W.E., (1968), "Logic Design with Integrated Circuits" (Book), p 74, John Wiley and Sons.

113. Younker, E.L., et al (1964), "Design of an Experimental Multiple Instantaneous Response File", Proc. AFIPS, SJCC Vol. 25, p 515.

114. Zadeh, L.A., "FUZZY sets", Information and Control, Vol. 8 p 338.

115. - (1970), "Intelligent Display System", Product Spot Light Datamation, Vol. 16, October, p 83.

ABBREVIATIONS USED

CACM:	Communication of ACM
JACM:	Journal of ACM
SJCC:	Spring Joint Computer Conference
FJCC:	Fall Joint Computer Conference

APPENDIX I

Distance Measure

The similarity measure defined in Section 3.3, namely,

$$\text{sim}\ (d_1, d_2) = \sum_{i=1}^{n} \frac{1}{|d_{1i} - d_{2i}|} \quad \text{for } d_{1i} \neq d_{2i}$$

$$= 0 \quad \text{otherwise}$$

is non negative and symmetric. But, the triangle inequality need not be always satisfied. For example, let

$$d_1 = <5>$$

$$d_2 = <10>$$

$$d_3 = <6>$$

Then

$$\frac{1}{|10-5|} + \frac{1}{|10-6|} \geq \frac{1}{|6-5|} \quad \text{is } \textbf{false}$$

Consider the following distance measure:

Distance between A and B; $d(A,B) = |A_r - B_r|$ where A_r and B_r are the relevance numbers (real numbers) assigned to the documents A and B with respect to the query under consideration. In its simplest form this could be the "grade of membership" of the documents as discussed in Section 3.5.

This measure defines a distance between two documents in the light of the given query. Considering the set of documents as domain, the "membership function" (See Section 3.5) maps them into the real line interval [0,1]. The above distance measure is defined over the range ratherthan the domain. It is to be noted that the mapping involved is not "one-to-one".

There can be more than one document giving rise to the same grade of membership for the given query. However, as far as that particular query is concerned such documents are equivalent or equivally important.

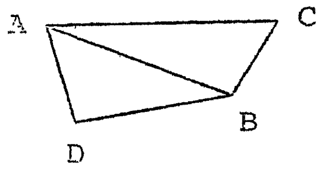
A number of other similarity measures satisfying the metric properties have been discussed by Salton (96). All such measures define the distance between two documents independent of the query.

APPENDIX II

n - level indirect relatedness

An n-level indirect relatedness becomes weaker as n increases:

Consider the three documents A, B, C.



By theorem 3

$$d(A,C) \leq d(A,B) + d(B,C)$$

where d(.) is the "distance" measure. Add a fourth document D as shown in the figure. Two level indirect relatedness between A and C is obtained from $d_2(A,C)$, the distance between A and C through D and B. Then

$$d_2(A,C) \leq d(A,D) + d(D,B) + d(B,C)$$

as

$$d(A,B) \leq d(A,D) + d(D,B)$$

$$d_2(A,C) \geq d(A,C).$$

The equality holds when all the documents are similar. Otherwise the distance between A and C is more when the reference is through B and D than the distance through B alone. As distance increases the relatedness would decrease.

For n level indirect-relatedness the same argument can be extended by including the remaining levels.

APPENDIX III

PMS and ISP notations

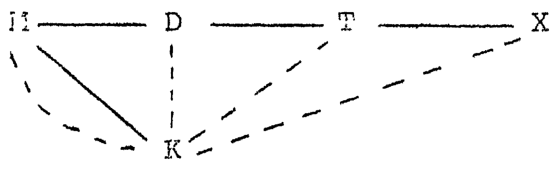
PMS Notation

This notation has been developed to describe the computer structure at the information flow level. The seven basic components of this system are described below:

1. Memory (M): A component used to store information units (i-units). The i-units are not changed in any way when they are stored in memory.
2. Link (L): A component that transfers i-units from one place to another in a computer system. Again there is no change in the i-units with this component.
3. Control (K): A component which evokes the operations of other components. Except the processor, P, all other components need some control to set them for work.
4. Switch (S): The switch is associated with a set of links, and can set some of them and break the rest.
5. Transducer (T): The transducer changes the i-units to encode a given meaning. For example the I/O device is a transducer.
6. Data Operation (D): This component produces i-units with new meaning, e.g., arithmetic, logic and shifting operations.

7. Processor (P): A component capable of interpreting a program and executing it. It consists of the components already mentioned.

As an example the classical diagram for Von Neumann computers is shown below in PMS notation



——— Information path X: external environment

- - - - Control signal path

<u>ISP Notation</u>

The ISP description provides a scheme for specifying any set of operations in a processor and any rules of interpretation. These two combindly describe the behaviour of a processor. The set of all operations in a processor are classified into two parts. One part contains those necessary to operate the D component in the PMS level and the other to operate links, switches, memories transducers etc. This is the level in which the programmer would like the processor to be described. All data operations are characterized as working on various data types. For example, signed or unsigned integers, character strings and real numbers are some data types.

The instruction set is described by the instruction expressions which is of the form:

Condition $\rightarrow$ Action-sequence

When the condition is satisfied the action sequence is evoked. Each action on a sequence has the form:

Memory expression $\leftarrow$ data expression

The left arrow indicates the transmit operations. The left-side specifies the memory location and the right side describes the information pattern to be placed. Each action in the sequence may itself be conditional, of the form: condition $\rightarrow$ action-sequence. Secondly, any sequential order or actions is indicated by the word "next . For instance

(A $\leftarrow$ B, B $\leftarrow$ A) would swap their contents

(A $\leftarrow$ B: next B $\leftarrow$ A) leaves A and B with the contents of B.

In ISP notation the processor state is described by providing names for the various bits in the register:

A <0:17> Accumulator.

This denotes an accumulator with 18 bits addressed as 0,1,...,17. Similarly, other indicators and special registers in the C.P.U. are described. The memory state is described by assigning name to each addressable unit (word or character) and to each bit in this i-unit.

M_p [0:3777_8] <0:17> denotes a memory M_p with 2K words and each word has 18 bits. The letter M stands for memory and the suffix distinguishes from other memory units. A number of notations are used in describing the action sequence like: $\sqcup$ for concatenation, $\neg$ for NOT, $\wedge$ for AND and so on. A summary of the notations and examples of many computer system described in these notations are available in the book cited earlier.

APPENDIX IV

SIMULATOR LISTING

```
$IBFTC MAIN
       LOGICAL FIRST
       COMMON /XX/ ERP,SCLCK,TCUST,BALK,CUSTP,TOTF
       COMMON /X1/ FIRST,NRUNS,IFIG
       COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
       COMMON /X20/ NR,IPNCH,IPRNT IOPT
       COMMON /X15/ QLMAX,QLMIN, [illegible](50,10),NIF,NCH,NOMAX,ANIF,NQM
       COMMON /X16/ OTCHCL(15),OTCHNT(15).OTCHLD(15),NOCH,
      1 SMAX,SMIN
       COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPCI,N,TCUSTP
       COMMON /X25/ ISP
       COMMON /X53/ IPC
       DIMENSION PS(4,10)
       IPC=1
C
C*****  IF IPC IS ZERO YOU GET COMP STAT PRINTED IN TABULAR FORM
C
       REWIND 2
       IPRNT=1
       FIRST=.TRUE.
       ISP=1
       REWIND 1
       READ 200,NST,MB,MF,MS,NB,NF,NS
       DO 20 I =MB,MF,MS
       NOCH=I
       PAUSE 1111
       DO 20 J=NB,NF,NS
       NCH=J
       NR=0
 14    NR=NR+1
       N=NST*(NR-1)+1
       IF(N .LE. NCH) NQM=N
       IF(NCH .LE. N) NQM=NCH
 200   FORMAT(7I3)
       IF(IPRNT .EQ. 1) PRINT 10,NR
 10    FORMAT(1H1,40X,*RUN NUMBER =*,I4/40X,13(1H*)//)
       PRINT 201,NR,N,NCH,NOCH,NQM
 201   FORMAT(5X,8I3)
       CALL DATAIN
       ISP=0
       IF(ERR.LT.0.0) STOP
       CALL SIMUL
       FIRST=.FALSE.
       IF(IOPT .EQ. 1) PRINT13,NR
 13    FORMAT(20X,*RUN *,I3,* COMPLETED*/)
 29    IF(NR .GE. NRUNS) GO TO 15
       GO TO 14
 15     IF(IPNCH .EQ. 1) PUNCH 16,XM1D,XM2D,XM3D,XM4D,XM5D
 16     FORMAT(5O12)
        IF(IOPT .EQ. 1) CALL PAROPT
 20     CONTINUE
        REWIND 2
        IF(IPC .EQ. 0) STOP
        DO 41 I=MB,MF,MS
        DO 41 J=NB,NF,NS
        DO 42 IK=1,4
        READ (2,43) (PS(IK,JK),JK=1,NRUNS)
 42     PUNCH 44,(PS(IK,JK),JK=1,NRUNS),I,J
 41     CONTINUE
 43     FORMAT(8E10.3)
 44     FORMAT(4E10.3,2X,2I3)
        CALL EXIT
        END
$IBFTC DATAIN
        SUBROUTINE DATAIN
        DIMENSION NAME(5),JTP(8)
        LOGICAL FIRST
        COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
        COMMON /X1/ FIRST,NRUNS,IFLG
        COMMON /X2/ ITYPE,TFIN,TASK
        COMMON /X3/ TARV,TBET
        COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPER,N,TCUSTP
        COMMON /X7/ SUMA(20,5)
        COMMON /X8/ SSUMA(20,5)
        COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
        COMMON /X11/ PARAM(15,9),FREQ(15,10)
        COMMON /X14/ CHCLCK(50),CHIDLE(50),CHLOAD(50),QL(50),
      1 QB(50)
        COMMON /X15/ QLMAX,QLMIN,QM(50,10),NIF,NCH,NQMAX,ANIF,NQM
        COMMON /X16/ OTCHCL(15),OTCHWT(15),OTCHLD(15),NOCH,
      1 SMAX,SMIN
        COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
        COMMON /X20/ NR,IPNCH,IPRNT,IOPT
        COMMON /X21/ ITPR,PRJ,PRT,PRJP,PRTP
        COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
      1 WOPS(130)
        COMMON /X25/ ISP
        IF(ISP .EQ. 0) GO TO 140
        IF(.NOT. FIRST) GO TO 14
C
C****** READ TYPE O CARDS FOR IDENTIFICATION
C
        READ 11,NRUNS,NPROB,ID,MON,IYR,NAME
 11     FORMAT(I2,4(1X,I2),5A6)
        PRINT 12
 12     FORMAT(/2X,120(1H*)/)
        PRINT 13,NAME,ID,MON,IYR,NPROB
 13      FORMAT(15X,5A6,3(I3,1H/),5X,*PROB.NO. *,I3)
         PRINT 12
 14      READ 16,(JTP(I),I=1,9),IFLG,IPNCH,IPRNT,IOPT
C****** ALL VARIABLES ARE INITIALIZED
C** *** IFLAG IS SET TO ONE FOR PE INITIALIZATION
C
 16      FORMAT(40I2)
         IF(IFLG .NE. 1) GO TO 15
 140     CONTINUE
         SCLCK=0.
         PRJP=0.
         PRTP=0.
         BALK=0.
         ERR=0.
         TCUST=0.
         TOTF=0.
         CUSTP=0.
         TCUSTP=0.
         QLMAX=0.
         QLMIN=1.0E 38
         NQMAX=10
         DO 50 I=1,15
         DO 49 J=1,9
 49      PARAM(I,J)=0.
         DO 50 J=1,10
 50      FREQ(I,J)=0.
         DO 60 I=1,20
         DO 60 J=1,5
         SUMA(I,J)=0.
 60      SSUMA(I,J)=0.
         DO 73 I=1,20
         SUMA(I,4)=1.0E 38
         SSUMA(I,4)=1.0E 38
 73      CONTINUE
         DO 61 I=1,50
         CHLOAD(I)=0.
         CHCLCK(I)=0.
         CHIDLE(I)=0.
         QB(I)=0.
         QL(I)=0.
         DO 61 J=1,10
         QM(I,J)=-1.0
 61      CONTINUE
         DO 68 I=1,15
         OTCHCL(I)=0.
         OTCHWT(I)=0.
         OTCHLD(I)=0.
 68      CONTINUE
         DO 62 I=1,130
         PBUS(I)=0.
```

```
      WORK(I)=0.
      WIPS(I)=0.
      WCPS(I)=0.
      TJOB(I)=0.
 62   CONTINUE
      IF( ISP .NE. 0) GO TO 15
      CALL SNDY1 (XM1D)
      CALL SNDY2 (XM2D)
      CALL SNDY3 (XM3D)
      CALL SNDY4 (XM4D)
      CALL SNDY5 (XM5D)
      RETURN
C
C** *** READ TYPE I DATA
C
 15   IF(JTP(1).EQ.0) GO TO 20
      READ 44,N,NOM
 44   FORMAT(2I4)
 37   FORMAT(/20X,*NO. OF PROCESSORS =*,I8 /)
      PRINT 37,N
C
C** *** READ TYPE II DATA
C
 20   IF(JTP(2).EQ.0) GO TO 25
      READ 36,XMA,XMP,XMR,XMS,XMT,PPEP
      PRINT 18,XMA,XMP,XMR,XMS,XMT
 18   FORMAT(10X,*MEAN INTER ARRIVAL TIME =*,F10.4,5X,
     1 *MEAN PROCESS TIME =*,F10.4/12X,*MEAN PRIORITY RATE =*,
     2 F10.4,5X,*MEAN STORAGE ACC.TIME =*,F10.4/)
       XMA=1.0/XMA
C
C** *** READ TYPE III CARD
C
 25   IF(JTP(3) .EQ. 0) GO TO 26
      READ 21,ITYPE,TFIN,TASK
 21   FORMAT(I2,3F8.3)
      IF(ITYPE.EQ.1) PRINT 22,TASK
      IF(ITYPE.EQ.2) PRINT 23,TFIN
 22   FORMAT(20X,*SIMULATION CONTROL PARAM. JOBS =*,F12.4/)
 23   FORMAT(20X,*SIMULATION CONTROL PARAMETER TIME =*,F12.4/)
C
C** *** READ TYPE  IV CARS
C
 26   IF(JTP(4) .EQ. 1) GOTO 30
      ITPR=0
      GO TO 27
 30   READ 21,ITPR,PRT,PRJ
C
C**-*** READ TYPE V CARDS
C
 27   IF(JTP(5) .EQ. 0) GO TO 40
```

```
      READ 38,XMF,NIF,NCH
 38   FORMAT(F8.3,2I4)
      ANIF=NIF
      PRINT 43,NIF,NCH,XMF
 43   FORMAT(/20X,*NO. OF FILES IN SYSTEM =*,I7,5X,*NO.OF*,
     1 * CHANNELS =*,I7,/23X,*MEAN NO. OF FILES REQUESTED FOR*,
     2 *SEARCH =*,F10.3//)
      XMF=1.0/XMF
C
C****** READ TYPE VI CARDS SEEDS FOR RANDOM GENERATORS
C
 40   IF(JTP(6) .EQ. 0) GO TO 41
 46   FORMAT(5O12)
      IF(IFLG .NE. 1) GO TO 41
      IF(FIRST) READ 46,XM1D,XM2D,XM3D,XM4D,XM5D
      CALL SNDY1(XM1D)
      CALL SNDY2(XM2D)
      CALL SNDY3(XM3D)
      CALL SNDY4(XM4D)
      CALL SNDY5(XM5D)
C
C****** TYPE VII DATA OUTPUT CHANNEL INFORMATION
C
 41   IF(JTP(7) .EQ. 0) GO TO 42
      READ 65,NOCH,XMN,XMX
 65   FORMAT(I2,2F8.3)
      PRINT 67,NOCH
 67   FORMAT(/20X,*NO. OF OUTPUT CHANNELS =*,I5/)
      PRINT 99,XMN,XMX
 99   FORMAT(10X,*OUTPUT TIME =*F10.3,2X,*TO*,F10.3,*PERCENT*/)
      SMAX=XMX*XMP
      SMIN=XMN*XMP
C
C****** TYPE VIII DATA
C
 42   IF(JTP(8) .EQ. 0) GO TO 99
      PRINT 32
 32   FORMAT(/30X,*INTERVAL LIMITS FOR HISTOGRAMS*//8X,*CODE*,
     1 20X,*RIGHT OPEN INTERVAL LIMITS*/)
      READ 16,NHIST
      IF(NHIST .GT. 15) NHIST=15
      DO 35 I=1,NHIST
      READ 36,(PARAM(I,J),J=1,9)
 36   FORMAT(10F8C3)
      PRINT 33,I,(PARAM(I,J),J=1,9)
 33   FORMAT(10X,I3,9(2X,F9.3))
 35   CONTINUE
 99   RETURN
       END
```

```
$IBFTC SIMUL
       SUBROUTINE SIMUL
       COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
       COMMON /X1/ FIRST,NRUNS,IFLG
       COMMON /X2/ ITYPE,TFIN,TASK
       COMMON /X3/ TARV,TBET
       COMMON /X14/ CHCL+K(50),CHIDLE(50),CHLOAD(50),QL(50),
     1 QB(50)
       COMMON /X21/ ITPR,PRJ,PPT,PRJP,PRTP
       COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
     1 WOPS(130)
 10    CALL ARRVL (TINT,JPR,2)
       CALL GENFIL
       TBET=TINT
       SCLCK=SCLCK+TINT
       TARV=SCLCK
       IF(ITPR.EQ.0) GO TO 15
       IF(ITPR .EQ. 1) GO TO 18
C
C****** CODE 1 IS PRINTING BY JOB CONTROL
C
 17    FORMAT(/15X,*SAMPLE OUTPUT AFTER *,E15.3,* UNITS OF*,
     1 * TIME* /)
       IF((SCLCK-PRTP).LT.PRT) GO TO 15
       PRTP=SCLCK
       PRINT 17,SCLCK
 20    CALL OUTPUT
       GO TO 15
 18    IF((TCUST-PRJP) .LT. PRJ) GO TO 15
       PRJP=TCUST
       PRINT 19,TCUST
 19    FORMAT(/15X,*SAMPLE OUTPUT AFTER *,F15.3,* ARRIVALS*/)
       GO TO 20
 15    IF(JPR.EQ.1) CUSTP=CUSTP+1.
       CALL PROCES
       IF(ERR .LT. 0.) RETURN
       IF(ITYPE.EQ.1) GO TO 12
       TMAX=YMAX(SCLCK,CHCLCK,PBLS)
C
C****** SIMULATION ENDS BY TIME CONTROL
C
       IF(TMAX.LT.TFIN) GO TO 10
 11    IF(SCLCK.GE.TFIN) GO TO 13
       CALL ARRVL (TINT,JPR,1)
       SCLCK=SCLCK+TINT
       BALK=BALK+1.
       GO TO 11
C
C****** SIMULATION ENDS BY NO. OF JOBS PROCESSED
C
```

```
 12     IF(TCUST .LT. TASK) GO TO 10
C
C****** UPDATE REQUIRED VARIABLES FOR THE LAST TIME SEGMENT
C
 13     CONTINUE
        CALL OUTPUT
        RETURN
        END
$IBFTC PROCES
        SUBROUTINE PROCES
        COMMON /XX/ ERR,SCLCK,TCUST,DALK,CUSTP,TOTF
        COMMON /X3/ TARV,TBET
        COMMON /X5/ XMA,XMP,XMR,XM,XMT,FRED,N,TCUSTP
        COMMON /X15/ QLMAX,QLMIN,QL(50,10),NIF,NCH,NOMAX,ANIF,NOM
        COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
        COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
      1 WOPS(130)
        CALL SCHED
        DO 13 I=1,NCH
        JFR=NPF(I)
        IF(JFR .GT. 0) GO TO 11
        CALL ERROR (60,0)
        GO TO 13
 11     XIDLE=0.
        WAIT=PBUS(JFR)-SCLCK
C
C****** TIME SPENT IN SCHEDULING (OVER HEAD) IS NOT ACCOUNTED
C
        IF(WAIT .GE. 0.) GO TO 12
        XIDLE=-WAIT
        WAIT=0.
 12     CALL COLCT (WAIT,1)
        CALL HISTO (WAIT,1)
C
C****** CODE 1 IS WAITING TIME OF JOBS FOR PROCESSORS
C
        PBUS(JFR)=PBUS(JFR)+XIDLE
 13     CONTINUE
        CALL DRUMAC
        IF(ERR .LT. 0.) RETURN
        CALL OUTBUF
C
C****** TO FIND THE TIME OF COMPLETION OF THIS JOB
C
        J=1
        JFR=NPF(J)
        TCOMP=PBUS(JFR)
        DO 15 I=1,NCH
        JFR=NPF(I)
        IF(JFR .GT. 0) GO TO 18
        CALL ERROR (61,0)
        GO TO 15
 18     IF(TCOMP .GE. PBUS(JFR)) GO TO 15
        TCOMP=PBUS(JFR)
 15     CONTINUE
        TRAN=TCOMP-TARV
        CALL COLCT (TRAN,2)
        CALL HISTO (TRAN,2)
        RETURN
        END
$IBFTC GENFIL
        SUBROUTINE GENFIL
        COMMON /XX/ ERR,SCLCK,TCUST, ALK,CUSTP,TOTF
        COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
        COMMON /X15/ QLMAX,QLMIN,CH(50,10),NIF,NCH,NQMAX,ANIF,NQM
        COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
        TIM=0.5
        ANCH=NCH
        DO 10 I=1,NCH
 10     MFJC(I)=0
        CALL DRAND (SEED,RNUM,4)
        XM4D=SEED
C
C****** NO. OF FILES EXPONENTIALLY DISTRIBUTED WITH MEAN XMF
C
        IF(RNUM .GT. 0.) TIM=-1./XMF*ALOG(RNUM)
        NFJ=TIM+0.5
C
C****** ATLEAST ONE FILE IS NEEDED FOR A JOB
C
        IF(NFJ .EQ. 0) NFJ=1
        IF(NFJ.GT.NIF) NFJ=NIF
        ANFJ=NFJ
        TOTF=TOTF+ANFJ
        I=0
 12     I=I+1
        CALL DRAND (SEED,RNUM,3)
        JC=RNUM*ANCH+0.5
        IF(JC .EQ. 0) JC=1
        IF(JC .GT. NCH) JC=NCH
        MFJC(JC)=MFJC(JC)+1
C
C****** NFJ FILES ARE GENERATED WITH UNIFORM DISTRIBUTION
C****** WHERE NCH IS THE NO. OF PHYSICAL FILE IN T+E SYSTEM
C
        IF(I .LT. NFJ) GO TO 12
        RETURN
        END
```

```
$IBFTC SCHED
       SUBROUTINE SCHED
       DIMENSION INDP(130),REM(130)
       COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
       COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPER,N,TCUSTP
       COMMON /X15/ QLMAX,QLMIN,QM(50,10),NIF,NCH,NQMAX,ANIF,NQM
       COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
       COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
     1 WOPS(130)
       DO 12 I=1,N
 12    REM(I)=PBUS(I)-SCLCK
C
C****** TO ARRANGE THE PROCESSORS IN THE INCREASING ORDER OF
C****** THEIR LEFT OVER BUSY PERIOD IN SERVICING EARLIER CALLS
C
       DO 14 I=1,N
       KJ=1
       SML=REM(KJ)
       DO 13 J=1,N
       IF(REM(J) .GE. SML) GO TO 13
       SML=REM(J)
       KJ=J
 13    CONTINUE
       INDP(I)=KJ
       REM(KJ)=1.0E38
 14    CONTINUE
C
C****** NQM PROCESSORS ARE ALLOTED PER JOB
C
       IC=1
 15    IXP=1
 17    NPF(IC)=INDP(IXP)
       IC=IC+1
       IXP=IXP+1
       IF(IC .GT. NCH) GO TO 18
       IF(IXP .GT. NQM) GO TO 15
       GO TO 17
 18    RETURN
       END
$IBFTC DRUMAC
       SUBROUTINE DRUMAC
       COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
       COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPER,N,TCUSTP
       COMMON /X14/ CHCLCK(50),CHIDLE(50),CHLOAD(50),QL(50),
     1 QB(50)
       COMMON /X15/ QLMAX,QLMIN,QM(50,10),NIF,NCH,NQMAX,ANIF,NQM
       COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
       COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
     1 WOPS(130)
```

```
      DO 11 I=1,NCH
      DO 11 J=1,NQMAX
      IF(QM(I,J) .LT. 0.) GO TO 11
      IF(QM(I,J) .GT. SCLCK) GO TO 11
      IF(QL(I) .GT. 0.) QL(I)=QL(I)-1.
      QM(I,J)=-1.
 11   CONTINUE
C
C****** DRUMAC EXPECTS PBUS TO HOLD THE TIME AT WHICH REQUEST
C****** FOR FILE IS MADE
C
      DO 16 I=1,NCH
      IF(MFJC(I).EQ. 0) GO TO 16
      AMF=MFJC(I)
C
C*****  ALL LOGICAL FILES IN ONE PHYSIC L FILE IS ALLOTED TO
C*****  ONE P E AND AMF IS NO OF LOGICAL FILES PER PHYSICAL FILE
C
      RIDLL=0.
      CALL OVHEAD (TM)
      TIME=XMS+TM
      DO 12 J=1,N-MAX
      IF(QM(I,J) .GE. 0.) GO TO 12
      JF=J
      GO TO 13
 12   CONTINUE
      QB(I)=QB(I)+1.
      JF=1
 13   IPR=NPF(I)
      WTM=CHCLCK(I)-PBUS(IPR)
      IF(WTM .GT. 0.) GO TO 14
      RIDLE=-WTM
      CHIDLE(I)=CHIDLE(I)+RIDLE
      WTM=0.
      GO TO 15
 14   QL(I)=QL(I)+1.
C
C****** WAITING TIME DOES NOT INCLUDE SERVICE TIME
C
 15   QM(I,JF)=CHCLCK(I)
C
C****** THE SELECTED FILE KEEPS ON FILLING MA AND MB ALTERNATE
C**Q*** UNTIL ALL THE AMF FILES ARE PROCESSED
C
      ZM=TIME+XMP*AMF
      PBUS(IPR)=PBUS(IPR)+WTM+ZM
      CHCLCK(I)=CHCLCK(I)+RIDLE+ZM
      WORK(IPR)=WORK(IPR)+AMF*XMP
      WIPS(IPR)=WIPS(IPR)+WTM
C****** COLLECTS THE NO OF PHYSICAL FILES
```

```
      TJOB(IPR)=TJOB(IPR)+1.0
      CHLOAD(I)=CHLOAD(I)+AMF
 16   CONTINUE
      DO 17 I=1,NCH
      IF(QLMAX .LT. QL(I)) QLMAX=QL(I)
      IF(QLMIN .GT. QL(I)) QLMIN=QL(I)
 17   CONTINUE
      QLEN=QLMAX
      CALL TMST (QLEN,5)
      CALL HISTO (QLEN,5)
      RETURN
      END
$IBFTC OUTBUF
      SUBROUTINE OUTBUF
      COMMON /X5/ XMA,XMP,XMP,XMU,XMT,PPER,N,TCUSTP
      COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
      COMMON /X16/ OTCHCL(15),OTCHWT(15),OTCHLD(15),NOCH,
     1 SMAX,SMIN
      COMMON /X15/ QLMAX,QLMIN,QM(50,10),NIF,NCH,NOMAX,ANIF,NQM
      COMMON /X17/ MFJC(50),NPF(50),NFJ,XMF
      COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
     1 WOPS(130)
      DIMENSION ALOC(100)
      SX=SMAX-SMIN
      NAL=1
      XMIN=OTCHCL(1)
      DO 12 I=1,NOCH
      IF(XMIN .LE. OTCHCL(I))  GO TO 12
      NAL=I
 12   CONTINUE
C
C****** ARRANGE PROCESSORS FOR SERVICE
C
      DO 13 I=1,NCH
      NR=NPF(I)
      IF(NR .EQ. 0) GO TO 14
      ALOC(I)=PBUS(NR)
      GO TO 13
 14   ALOC(I)=-1.0
 13   CONTINUE
C
C****** SERVICING THE PROCESSOR REQUESTS FOR OUTPUT
C
      DO 20 K=1,NCH
      XMIN=1.0E38
      NPR=1
      DO 15 I=1,NCH
      IF(ALOC(I) .LE. 0.0) GO TO 15
      IF(XMIN .LT. ALOC(I)) GO TO 15
      NPR=I
      XMIN=ALOC(I)
 15   CONTINUE
      ALOC(NPR)=-1.0
      XIDLE=0.
      WAIT=OTCHCL(NAL)-PBUS(NPR)
      IF(WAIT .GE. 0.) GO TO 17
      XIDLE=-WAIT
      WAIT=0.
 17   WOPS(NPR)=WOPS(NPR)+WAIT
      OTCHWT(NAL)=OTCHWT(NAL)+XIDLE
      CALL DRAND (SEED,RNUM,3)
      TIM=(SMIN+SX*RNUM)*FLOAT(MFJC(K))
      PBUS(NPR)=PBUS(NPR)+WAIT+TIM
      OTCHCL(NAL)=OTCHCL(NAL)+XIDLE+TIM
      OTCHLD(NAL)=OTCHLD(NAL)+TIM
 20   CONTINUE
      RETURN
      END
$IBFTC ARRVL
      SUBROUTINE ARRVL (TINT,JPR,K)
      COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
      COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPER,N,TCUSTP
      COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
      JPR=0
      TINT=0.
      CALL DRAND (SEED,RNUM,1)
      XM1D=SEED
      IF(RNUM .GT. 0.) TINT=-1./XMA*ALOG(RNUM)
      TCUST=TCUST+1.
      IF(K.EQ.1) RETURN
      NPSSN=0
      Y=EXP(-XMP)
      X=1.0
 3    CALL DRAND (SEED,RNUM,3)
      XM3D=SEED
      IF(X-Y) 6,7,7
 7    X=X*RNUM
      NPSSN=NPSSN+1
      GO TO 3
 6    PER=FLOAT(NPSSN)/(TCUST-TCUSTP)
      IF(PER .LE. PPER) RETURN
      TCUSTP=TCUST
      JPR=1
C
C****** SEE GASP BOOK PAGE 100 FOR THESE ALGORITHMS
C
      RETURN
      END
```

```
$IBFTC DRAND
       SUBROUTINE DRAND (SEED,RNUM,IG)
       COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
       X=1.0
C
C****** X IS DUMMY ARGUMENT WHICH IS IGNORED BY RANDOM GENS
C
       IF(IG .GT. 5) RETURN
       GO TO (1,2,3,4,5),IG
 1     RNUM=RNDY1(X)
       SEED=WNDY1(X)
       RETURN
 2     RNUM=RNDY2(X)
       SEED=WNDY2(X)
       RETURN
 3     RNUM=RNDY3(X)
       SEED = WNDY3(X)
 4     RNUM=RNDY4(X)
       SEED = WNDY4(X)
       RETURN
 5     RNUM=RNDY5(X)
       SEED=WNDY5(X)
       RETURN
       END
$IBFTC YMAX
       FUNCTION YMAX (SCLCK,CHCLCK,PBUS)
       DIMENSION CHCLCK(50),PBUS(130)
       YMAX=SCLCK
       DO 2 I=1,50
       IF(YMAX .GE. CHCLCK(I)) GO TO 2
       YMAX=CHCLCK(I)
 2     CONTINUE
       DO 3 I=1,130
       IF(YMAX .GE. PBUS(I)) GO TO 3
       YMAX=PBUS(I)
 3     CONTINUE
       RETURN
       END
$IBFTC ERROR
       SUBROUTINE ERROR (NCODE,KFLG)
       COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
       PRINT 2,NCODE
 2     FORMAT(/30X,*ERROR TYPE *,I4/)
       IF(KFLG.EQ.0) RETURN
C
C****** IRRECOVERABLE ERROR TURN ON TERMINATION FLAG
C
       ERR=-1.
       PRINT 3
 3     FORMAT(/25X,*DUMP OF SYSTEM STATUS */23X,28(1H-)//)
```

```
        CALL OUTPUT
        RETURN
        END
$IBFTC OVHEAD
        SUBROUTINE OVHEAD (TM)
        COMMON /X5/ XMA,XMP,XMR,XMS,XMT,PPER,N,TCUSTP
        COMMON /X13/ XM1D,XM2D,XM3D,XM4D,XM5D
        COMMON /X15/ QLMAX,QLMIN,CM(50,10),NIF,NCH,NQMAX,ANIF,NQM
C
C****** ROTARY FILE CONCEPT IS ASSUMED
C
        TM=0.
        IF(NIF .LE. NCH) RETURN
        CALL DRAND (SEED,RNUM,2)
        XM2D=SEED
        TM=RNUM*XMS*0.25
C
C***** 25 PERCENT CHOSEN FROM PDP 11 DMS
C
        RETURN
        END
$IBFTC COLCT
        SUBROUTINE COLCT (X,KN)
        COMMON /X7/ SUMA(20,5)
        N=KN
        IF((N.GT.0) .AND. (N.LE.20)) GO TO 12
        CALL ERROR (11,0)
        RETURN
 12     SUMA(N,1)=SUMA(N,1)+X
        SUMA(N,2)=SUMA(N,2)+X*X
        SUMA(N,3)=SUMA(N,3)+1.
        SUMA(N,4)=AMIN1(SUMA(N,4),X)
        SUMA(N,5)=AMAX1(SUMA(N,5),X)
        RETURN
        END
$IBFTC HISTO
        SUBROUTINE HISTO (X,IC)
        COMMON /X11/ PARAM(15,9),FREC(15,10)
        IF(IC.LE. 15) GO TO 2
        CALL ERROR (111,0)
        RETURN
 2      IF(X.GE.PARAM(IC,1)) GO TO 3
        J=1
        GO TO 6
 3      DO 5 I=2,9
        IF(X .GE. PARAM(IC,I)) GO TO 5
        J=I
        GO TO 6
C
C****** CLASS INTERVALS LEFT CLOSED AND RIGHT OPENIN INCREASING
C****** ORDER AS MANY AS 9 CAN BE GIVEN.TOTAL 10 CLASSES
```

```
 5      CONTINUE
        J=10
 6      FREQ(IC,J)=FREQ(IC,J)+1.0
        RETURN
        END
$IBFTC  TMST
        SUBROUTINE TMST  (X,N)
        COMMON /XX/  ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
        COMMON /X8/  SSUMA(20,5)
        IF((N.GT.0)  .AND.  (N.LE.20))  GO  TO  20
        CALL  ERROR  (12,0)
        RETURN
 20     TT=SCLCK-SSUMA(N,1)
        SSUMA(N,1)=SCLCK
        SSUMA  (N,2)  =  SSUMA  (N,2)  + X  *  TT
        SSUMA(N,3)=SSUMA(N,3)+X*X*TT
        SSUMA(N,4)=AMIN1(SSUMA(N,4),X)
        SSUMA(N,5)=AMAX1(SSUMA(N,5),X)
        RETURN
        END
$IBFTC  STAT
        SUBROUTINE  STAT  (IC,JC)
        COMMON  /X7/  SUMA(20,5)
        COMMON  /X8/  SSUMA(20,5)
        COMMON  /X11/  PARAM(15,9),FREQ(15,10)
        COMMON  /X20/  NR,IPNCH,IPRNT,IOPT
C
C******  JC  IS  0  FOR  COLCT  STAT  AND  1  FOR  TMST  STATISTICS
C
        IF(JC.EQ.1)  GO  TO  12
        IF(SUMA(IC,3).LE.0.)  GO  TO  17
        AVER=SUMA(IC,1)/SUMA(IC,3)
        VAR=SUMA(IC,2)/SUMA(IC,3)-AVER**2
        SMALL=SUMA(IC,4)
        BIG=SUMA(IC,5)
        GO  TO  13
 17     AVER=0.
        VAR=0.
        BIG=0.
        SMALL  =  0.
        GO  TO  13
 12     AVER=SSUMA(IC,2)/SSUMA(IC,1)
        VAR=SSUMA(IC,3)/SSUMA(IC,1)-AVER**2
        SMALL=SSUMA(IC,4)
        BIG=SSUMA(IC,5)
 13     IF(IPRNT .EQ. 1)  PRINT  11,AVER,VAR,BIG,SMALL
 11     FORMAT(/20X,*MEAN  =*,E12.5,5X,*VARIANCE  =*,A12.5/
      1 20X,*MAXIMUM  =*,E12.5,5X,*MINIMUM  =*,E12.5/)
        IF(IOPT.EQ.1)  WRITE(1,16)  AVER,VAR,BIG,SMALL
 16     FORMAT(8E12.5)
```

```
          IF(IPRNT .EQ. 0) RETURN
          PRINT 15
   15     FORMAT(//25X,*DISTRIBUTION*//)
          PRINT 14,(FREQ(IC,J),J=1,10)
   14     FORMAT(2H /,10(F10.2,2H  /)/)
          RETURN
          END
$IBFTC PAPOPT
          SUBROUTINE PAPOPT
          COMMON /X1/ FIRST,NRUNS,IFLG
          COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
        1 WOPS(130)
          EQUIVALENCE (PBUS,PAR)
          COMMON /X53/ IPC
          DIMENSION PAR(10,60),PS(4),JXR(4)
          DATA JXR/1,9,25,41/
          NUN=1
          REWIND NUN
          DO 13 I=1,NRUNS
          DO 12 J=1,12
          JS=(J-1)*4+1
          JF=JS+3
   12     READ(NUN,14)(PAR(I,K),K=JS,JF)
   14     FORMAT(8E12.5)
          JS=JF+1
          JF=JF+8
   13     READ(NUN,14) (PAR(I,K),K=JS,JF)
          PRINT 60
   60     FORMAT(///10X,*1.TRANSIT TIME  2.CHANN UTL  3.PROC UTL *
        1 *  4.OTCHCL UTL*/)
          DO 61 I=1,NRUNS
          DO 62 J=1,4
          JK=JXR(J)
   62     PS(J)=PAR(I,JK)
          WRITE(2,32) (PS(J),J=1,4)
          IF(IPC .EQ. 1) PRINT 63,(PS(J),J=1,4)
   63     FORMAT(10X,4(E12.4,3X))
   61     CONTINUE
   32     FORMAT(5F10.3)
          PRINT 15
   15     FORMAT(1H1,30X,*JOBSTATISTICS*//35X,*TRANSIT TIME*,40X,
        1 *WAITING TIME*/)
          PRINT 16
   16     FORMAT(2(14X,*MEAN*,10X,*VARIAN*,9X,*MAXIM*,10X,*MINIM*,
        1 )/)
          DO 17 I=1,NRUNS
   17     PRINT 18,I,(PAR(I,J),J=1,8)
   18     FORMAT(5X,I3,4E14.5,5X,4E14.5)
          PRINT 19
   19     FORMAT(1H1,30X,*CHANNEL STATISTICS*//35X,*UTILIZATION*,40X,
        1 *LOAD SHARING*/)
```

```
      PRINT 16
      DO 20 I=1,NRUNS
 20   PRINT 18,I,(PAR(I,J),J=9,16)
      PRINT 21
 21   FORMAT(///35X,*CHANNEL QUEUE*,40X,*OM OVERFLOW */)
      PRINT 16
      DO 22 I=1,NRUNS
 22   PRINT 18,I,(PAR(I,J),J=17,24)
      PRINT 23
 23   FORMAT(1H1,30X,*PROCESSOR STATISTICS*//35X,*UTLIZATION*,
     1 40X,*LOAD SHARING*/)
      PRINT 16
      DO 24 I=1,NRUNS
 24   PRINT 18,I,(PAR(I,J),J=25,32)
      PRINT 25
 25   FORMAT(/// 35X,*WAITING FOR FILE*,40X,*WAITING FOR*,
     1 *BUFFER*/)
      PRINT 16
      DO 26 I=1,NRUNS
 26   PRINT 18,I,(PAR(I,J),J=33,40)
      PRINT 27
 27   FORMAT(1H1 //30X,*OUTPUT CHANNEL STATISTICS*//35X,
     1 *UTILIZATION*,45X,*LOAD SHARING* //)
      PRINT 16
      DO 28 I=1,NRUNS
 28   PRINT 18,I,(PAR(I,J),J=41,48)
      PRINT 29
 29   FORMAT(1H1 //3X,*1.TACS 2. SCLCK 3.BALK 4.SUM 5.QBMAX*,
     1 *6.QBMIN 7.QLMAX 8.QLMIN *//)
      DO 30 I=1,NRUNS
 30   PRINT 18,I,(PAR(I,J),J=49,56)
      RETURN
      END
$IBFTC OUTPUT
      SUBROUTINE OUTPUT
      COMMON /XX/ ERR,SCLCK,TCUST,BALK,CUSTP,TOTF
      COMMON /X5/ XMA,XMP,XME,XMS,XMT,PPER,N,TCUSTP
      COMMON /X14/ CHCL+K(50),CHIDLE(50),CHLOAD(50),QL(50),
     1 QB(50)
      COMMON /X15/ QLMAX,QLMIN,QA(50,10),NIF,NCH,NQMAX,ANIF,NCM
      COMMON /X16/ OTCHCL(15),OTCINT(15),OTCHLD(15),NOCH,
     1 SMAX,SMIN
      COMMON /X20/ NR,IPNCH,IPRNT,IOPT
      COMMON /X51/ PBUS(130),WORK(130),TJOB(130),WIPS(130),
     1 WOPS(130)
      IF(IPRNT .EQ. 1) PRINT 10
 10   FORMAT(26X,*JOB STATISTICS*/24X,18(1H-)//20X,*TRANSIT*,
     1 *TIME OF  JOBS*//)
      CALL STAT (2,0)
      IF(IPRNT .EQ. 1) PRINT 11
 11   FORMAT(20X,*WAITING TIME OF JOBS FOR A PROCESSOR*/)
      CALL STAT (1,0)
      QBT=0
```

```
          TACS=0.
          QBMAX=0.
          QBMIN=1.0E 38
          DO 15 I=1,NCH
          IF(QBMAX.LT. QB(I)) QBMAX=QB(I)
          IF(QBMIN .GT. QB(I)) QBMIN=QB(I)
          QBT=QBT+QB(I)
   15     TACS=TACS+CHLOAD(I)
C
C****** CHANNEL UTILIZATION STATISTICS
C
          IF(IPRNT.EQ.1) PRINT 23,NCH
   23     FORMAT(/25X,*CHANNEL UTILIZATION STATISTICS*/23X,34(1H-),
        1 /20X,*FOR NO. OF CHANNELS =*,I5//)
          DO 22 I=1,NCH
          QLT=0.
          UTL=(CHCLCK(I)-CHIDLE(I))*100./CHCLCK(I)
          CALL COLCT (UTL,3)
          CALL HISTO (UTL,3)
          PERL=100.0*CHLOAD(I)/TACS
          CALL COLCT  (PERL,4)
          CALL HISTO  (PERL,4)
          IF(QBT.GT 0.) QLT=QB(I)*100./QBT
          CALL COLCT (QLT,6)
          CALL HISTO (QLT,6)
   22     CONTINUE
          CALL STAT(3,0)
          IF(IPRNT.EQ.1) PRINT 24,TACS
   24     FORMAT(25X,*CHANNEL LOAD SHARING*/23X,24(1H-)/25X,
        1 *FOR NO. OF ACCESS =*,E12.3/)
          CALL STAT(4,0)
          IF(IPRNT .NE. 1) GO TO 30
          PRINT 12,QLMAX,QLMIN
   12     FORMAT(/25X,*INDEX FILE ACCESS STATISTICS */23X,30(1H-),
        1 //20X,*QUEUE STATISTICS*/30X,*ABSOLUTE MAXIMUM =*,E12.3
        2 /30X,*ABSOLUTE MINIMUM =*,E12.3//)
   30     CALL STAT (5,1)
          IF(IPRNT.EQ.1) PRINT 25,QBMAX,QLMIN
   25     FORMAT(/25X,*OVERFLOW IN Q  MATRIX*/23X,21(1H-)/20X,
        1 *ABSOLUTE MAXIMUM =*,E12.3/20X,
        2 *ABSOLUTE MINIMUM =*,E12.3//)
          CALL STAT(6,0)
C
C****** PROCFSSOR UTILIZATTON
C
          DO 16 I=1,N
          UTL=0.
          IF(PBUS(I) .GT. 0.) UTL=WORK(I)*100./PBUS(I)
          CALL COLCT UUTL,7)
          CALL HISTO UUTL,7)
          PERJ=TJOB(I)*100.0/TOTF
          CALL HISTO (PERJ,8)
```

```
      CALL COLCT (PERJ,8)
      WIPS(I)=WIPS(I)/TJOB(I)
      CALL COLCT (WIPS(I),9)
      CALL HISTO (WIPS(I),9)
      WOPS(I)=WOPS(I)/TJOB(I)
      CALL COLCT(WOPS(I),10)
      CALL HISTO (WOPS(I),10)
 16   CONTINUE
      IF(IPRNT .EQ. 1) PRINT 17
 17   FORMAT(/25XT*UTILIZATION OF PROCESSORS*/23X,28(1H-)///)
      CALL STAT(7,0)
      IF(IPRNT.EQ.1) PRINT 18,TOTF
 18   FORMAT(/25XT*LOAD SHARING OF PROCESSORS*/23X,28(1H-)//
     1 20X,*100 PERCENT CORRESPONDS TO *,E12.3,* FILES*/)
      CALL STAT(8,0)
      IF(IPRNT.EQ.1) PRINT 19
 19   FORMAT(/25X,*WAITING TIME DISTRIBUTION OF PROCESSORS*,
     1 /23X,50(1H-)///)
      CALL STAT(9,0)
      IF(IPRNT.EQ.1) PRINT 20
 20   FORMAT(/25X,*WAITING TIME FOR OUTPUT CHANNEL*/23X,35(1H-,
     1 )///)
      CALL STAT (10,0)
C
C****** OUTPUT CHANNEL STATISTICS
C
      SUM=0.
      DO 31 I=1,NOCH
 31   SUM=SUM+OTCHLD(I)
      DO 32 I=1,NOCH
      UTL=(OTCHCL(I)-OTCHWT(I))*100./OTCHCL(I)
      CALL COLCT (UTL,11)
      CALL HISTO (UTL,11)
      PERL=OTCHLD(I)*100./SUM
      CALL COLCT (PERL,12)
      CALL HISTO(PERL,12)
 32   CONTINUE
      IF(IPRNT .EQ. 1) PRINT 33
 33   FORMAT(/25X,*OUPUT CHANNEL UTILIZATION*/23X,30(1H-)///)
      CALL STAT (11,0)
      IF(IPRNT .EQ. 1) PRINT 34,SUM
 34   FORMAT(25X,*OUTPUT CHANNEL LOAD SHARING*/23X,30(1H-)//
     1 25X,*100 PERCENT CORRESPONDS TO  *,E12.5//)
      CALL STAT (12,0)
      IF(IOPT.EQ.1) WRITE(1,27) TACS,SCLCK,HALK,SUM,QBMAX,
     1 QBMIN,QLMAX,QLMIN
 27   FORMAT(8E12.5)
      PRINT 14,TACS,TCUST,SCLCK
 14   FORMAT(20X,*TOTAL FILE ACCESS =*,E12.4/
     1 23X,*FOR NO. OF CUSTOMERS =*,E12.4/
     2 26X,*OVER A PERIOD OF TIME =*,E12.4/)
      RETURN
      END
```

APPENDIX V

Sample Output

```
***************************************************************************
       RADHAKRISHNAN          8/6/71/          PROB. NO. 5
***************************************************************************

               NO. OF PROCESSORS =     4
      MEAN INTER ARRIVAL TIME = 30.0000  MEAN PROCESS TIME = 1.5000
        MEAN PRIORITY RATE    = 0.0100    MEAN STORAGE ACC. TIME = 3.0000
          MEAN OUTPUT TIME    = 10  TO 20 PERCENT

               SIMULATION CONTROL PARAMETER JOBS = 500.0000

               NO. OF FILES IN SYSTEM = 100  NO. OF CHANNELS =    3
                  MEAN NO. OF FILES REQUESTED FOR SEARCH = 50.000

               NO. OF OUTPUT CHANNELS = 3

                  INTERVAL LIMITS FOR HISTOGRAMS

        CODES          RIGHT OPEN INTERVAL LIMITS

 1    0.500  1.000  1.500  2.000  2.500  3.000  5.000  10.000   12.000
 2    1.000  2.000  3.000  4.000  5.000  6.000  7.000   8.000    9.000
 3    1.000  2.000  3.000  4.000  5.000  6.000  7.000   8.000    9.000
 4    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
 5    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
 6    1.000  2.000  3.000  4.000  5.000  6.000  7.000   8.000    9.000
 7    1.000  2.000  3.000  4.000  5.000  6.000  7.000   8.000    9.000
 8    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
 9    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
10    0.500  1.000  1.500  2.000  2.500  3.000  5.000  10.000   12.000
11    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
12    5.000 10.000 15.000 20.000 30.000 40.000 50.000  60.000   70.000
```

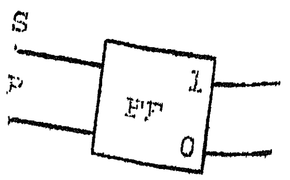
```
                    JOB STATISTICS
          TRANSIT TIME OF   JOBS

          MEAN = 0.73699E 02        VARIANCE = 0.30747E 04
          MAXIMUM = 0.31475E 03       MINIMUM = 0.46718E 01

                    DISTRIBUTION

/ 0.00 / 0.00 / 0.00/ 0.00 / 5.00 / 10.00 / 7.00 / 3.00 / 1.00 / 474.00 /

                    WAITING TIME OF JOBS FOR 1 PROCESSOR

                    MEAN = 0.28828E 02      VARIANCE = 0.20050E 04
                    MAXIMUM = 0.25558E 03    MINIMUM = 0.

                     DISTRIBUTION

/ 727.00 / 8.00 / 7.00 / 8.00 / 2.00 / 7.00 / 11.00 / 51.00 / 16.00 / 653.00 /

                    CHANNEL UTILIZATION STATISTICS

                    FOR NO. OF CHANNELS = 3

                    MEAN = 0.38326E 02      VARIANCE = 0.19424E 03
                    MAXIMUM = 0.55297E 02  MINIMUM = 0.21160E 02

                     DISTRIBUTION

/ 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 3.00 /

                    CHANNEL LOAD SHARING
                    FOR NO. OF ACCESS = 0.2171 05

                    MEAN = 0.33333E 02      VARIANCE = 0.18893E 03
                    MAXIMUM = 0.50099E 02    MINIMUM = 0.16431E 02

/ 0.00 / 0.00 / 0.00 / 1.00 / 0.00 / 1.00 / 0.00 / 1.00 / 0.00 / 0.00 /
```

```
                      UTILIZATION OF PROCESSORS

                  MEAN = 0.25060E 02      VARIANCE = 0.82303E 02
                  MAXIMUM = 0.33601E 02   MINIMUM  = 0.11998E 02

                            DISTRIBUTION

/0.00  / 0.00 / 0.00 / 0.00 / 0.00/ 0.00 / 0.00 / 0.00 / 0.00 / 4.00 /

                      LOAD SHARING OF PROCESSORS

                  100 PERCENT CORRESPONDS TO  0.217E 05 FILES

                  MEAN = 0.16261E 01      VARIANCE = 0.10665E 00
                  MAXIMUM = 0.19946E 01   MINIMUM  = 0.11101E 01

                            DISTRIBUTION

/ 4.00 / 0.00 / 0.00 / 0.00/ 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 /

                  WAITING TIME FOR INPUT CHANNEL

                  MEAN = 0.33214E 01      VARIANCE = 0.81054E 01
                  MAXIMUM = 0.77654E 01   MINIMUM  = 0.52622E 00

                             DISTRIBUTION

 / 3.00 / 1.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 /  0.00 / 0.00 / 0.00 /

                  WAITING TIME FOR OUTPUT CHANNEL

                  MEAN = 0.25940E 02      VARIANCE = 0.94125E 03
                  MAXIMUM = 0.75840E 02   MINIMUM  = 0.

                             DISTRIBUTION

/ 1.00 / 0.00 / 1.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 2.00 /
```

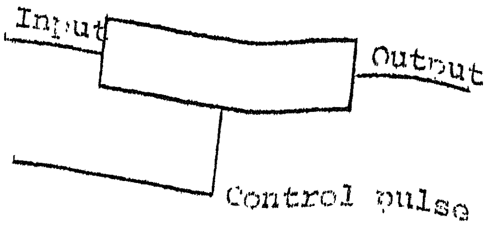
```
                          OUTPUT CHANNEL UTILIZATION

                          MEAN = 0.50487E 01     VARIANCE = 0.50086E-01
                          MAXIMUM = 0.52764E 01 MINIMUM  = 0.47445E 01

                          DISTRIBUTION

/ 1.00 / 2.00 / 0.00 / 0.00 / 0.00 / 0.00 / 0.00 /  0.00 / 0.00 / 0.00 /

                          OUTPUT CHANNEL LOAD SHARING

                          100 PERCENT CORRESPONDS TO 0.49105E 04

                          MEAN = 0.33333E 02     VARIANCE = 0.20719E 01
                          MAXIMUM = 0.34839E 02 MINIMUM  = 0.31394E 02

                          DISTRIBUTION

/ 0.00 / 0.00 / 0.00 / 0.00 / 0.00 / 3.00 / 0.00 / 0.00 / 0.00 / 0.00 /

                  TOTAL FILE ACCESS = 0.2171E 05
                    FOR NO. OF CUSTOMERS = 0.5000E 03
                      OVER A PERIOD OF TIME = 0.3234E 05

                  RUN  1  COMPLETED
```

APPENDIX VI

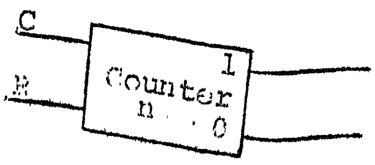
List of Symbols Used for Logic Diagrams

(1) 

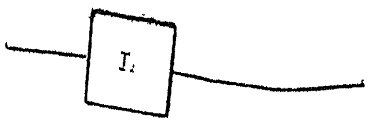
Flip Flop

(2) 

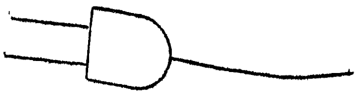
Serial Shift Register

(3) 

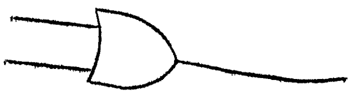
Level output ON when n clock pulses are given

(4) 

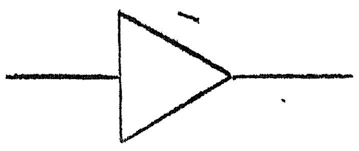
Latch (one shot multivibrator)

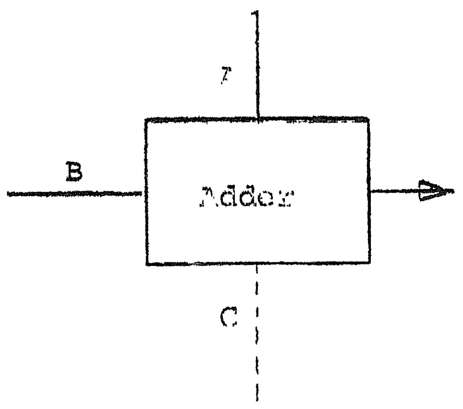
(5) 

AND gate

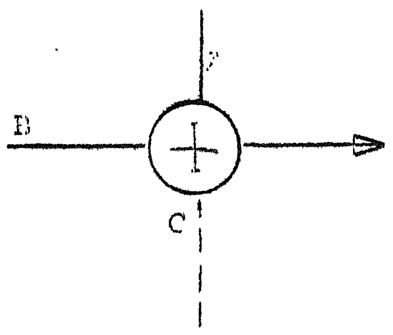
(6) 

OR gate

(7)  NOT gate

(8) 

A+B if C is ON
inhibited otherwise

(9) 

Exclusive OR gating.
[Controlled by C]

EE-1571-D-RAN-.Y.